

HAYDEN

Macintosh Library

BOOKS

046567

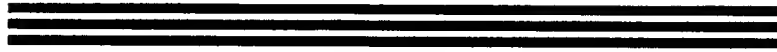
MacAccess

Information in Motion



D e a n G e n g l e
& S t e v e n S m i t h

MacAccess: Information in Motion



MacAccess: Information in Motion

Dean Gengle and Steven Smith

A CommuniTree Book

HAYDEN BOOKS

A Division of Howard W. Sams & Company

4300 West 62nd Street

Indianapolis, Indiana 46268 USA

Copyright © 1987 Hayden Books
A Division of Howard W. Sams, Inc.

FIRST EDITION
FIRST PRINTING 1987

All rights reserved. No part of this book shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein. While every precaution has been taken in the preparation of this book, the publisher and authors assume no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained herein.

International Standard Book Number: 0-672-46567-1
Library of Congress Catalog Card Number: 86-63337

Acquisitions Editor: BILL GROUT
Cover Design: JIM BERNARD
Compositor: THE PUBLISHER'S NETWORK, MORRISVILLE, PA

Printed in the United States of America

Dedication



This portion of the Great Work is dedicated to John S. James and John Mellen, whose advice, support, and achievements have inspired us ever onward.

Trademark Acknowledgments

All terms mentioned in this book that are known to be trademarks or service marks are listed below. In addition, terms suspected of being trademarks or service marks have been appropriately capitalized. Hayden cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

1-2-3 is a registered trademark of Lotus Development Corporation

Apple is a registered trademark of Apple Computer, Inc.

AppleTalk is a registered trademark of Apple Computer, Inc.

BLAST is a registered trademark of Communications Research Group

Chart is a registered trademark of Microsoft Corporation

COMPAQ is a registered trademark of Compaq Computer Corporation

CompuServe is a registered trademark of CompuServe Information Services, an H&R Block Company

Crosstalk is a registered trademark of Microstuf Inc.

dBASE is a registered trademark of Ashton-Tate

DIALOG is a registered trademark of Dialog Information Services, Inc.

Displaywriter is a trademark of International Business Machines Corporation

Dow Jones News/Retrieval is a registered trademark of Dow Jones & Company, Inc.

Excel is a trademark of Microsoft Corporation

IBM AT is a registered trademark of International Business Machines Corporation

inTalk is a registered trademark of Palantir Software

Jazz is a registered trademark of Lotus Development Corporation

Kaypro is a registered trademark of Kaypro Corporation

Kermit is a registered trademark of Henson Associates, licensed to Columbia University

LaserWriter is a registered trademark of Apple Computer, Inc.

MacCharlie is a registered trademark of Dayna Communications, Inc.

MacDraw is a trademark of Apple Computer, Inc.

MacGeorge is a registered trademark of RD Communications, Inc.

Macintosh is a trademark of Apple Computer, Inc.

MacLink is a registered trademark of DataViz, Inc.

MacMail is a registered trademark of Aegis Development, Inc.

MacMainframe is a registered trademark of Avatar Technologies, Inc.

MacPaint is a trademark of Apple Computer, Inc.

MacTerminal is a trademark of Apple Computer, Inc.

MacWrite is a trademark of Apple Computer, Inc.

Market Manager Plus is a registered trademark of Dow Jones & Company, Inc.

Maxwell Modem is a registered trademark of RD Communications, Inc.
MCI Mail is a trademark of MCI Communications Corp.
MicroModem II is a trademark of Hayes Microcomputer Products, Inc.
MicroPhone is a registered trademark of Software Ventures, Inc.
MultiMate is a registered trademark of MultiMate International Corporation, an Ashton-Tate Company
MultiPlan is a registered trademark of Microsoft Corporation
N'Cryptor is a registered trademark of Mainstay
Newsnet is a registered trademark of Independent Publications, Inc.
Osborne is a registered trademark of Osborne Computer Corporation
PageMaker is a registered trademark of Aldus Corporation
Paradox is a registered trademark of Ansa Software
PC to Mac and Back is a registered trademark of Dilithium Press
PC Teletree is a registered trademark of CommuniTree Group
ProModem is a registered trademark of Prometheus Products, Inc.
Popcom is a registered trademark of Prentice Corporation
Quick and Dirty Utilities is a registered trademark of Dreams of the Phoenix, Inc.
Racal-Vadic is a registered trademark of RD Communications, Inc.
ReadySetGo is a registered trademark of Manhattan Graphics
Red Ryder is a registered trademark of The Freesoft Company
Samna Word is a registered trademark of Samna Corporation
SideKick is a registered trademark of Borland International
Smart-Cat is a registered trademark of Novation, Inc.
SmartCable is a registered trademark of IQ Technologies
Smartmodem followed by model number is a trademark of Microcomputer Products, Inc.
Softstrip is a registered trademark of Cauzin Systems, Inc.
Spinwriter is a registered trademark of NEC Home Electronics, Inc.
Sprint is a registered trademark of US Sprint Communications
Straight Talk is a registered trademark of Dow Jones & Company, Inc.
Tektronix is a registered trademark of Tektronix, Inc.
Telescope is a registered trademark of Mainstay
TeleVideo is a registered trademark of Televideo Systems, Inc.
Tempo is a registered trademark of Affinity Microsystems, Ltd.
The Source is a service mark of The Source Telecomputing Corporation
TOPS is a registered trademark of Centram Systems West
Trailblazer is a registered trademark of Telebit Corporation
UNIX is a registered trademark of AT&T
VersaTerm is a registered trademark of Peripherals, Supplie & Computers
VisiCalc is a registered trademark of VisiCorp
WORD is a trademark of Microsoft Corporation
WordPerfect is a registered trademark of WordPerfect Corporation
WordStar is a registered trademark of MicroPro International Corp.
XENIX is a registered trademark of Microsoft Corporation

Contents



	Acknowledgments	xvii
	Introduction	xix
Chapter 1	Coming to Terms	1

	Softspace	1
	Files	2
	Modem	2
	Telephones and Phone	3
	Software	3
	Mainframes	4
	Micros	5
	Networks	5
	Information	5
Chapter 2	As the Modem Squeals	7

	Setting the Stage	7
	The Scenario	8
	Is Telecom Worth It?	11
	Questionnaire	13

Chapter 3	Digging into Telecom	17
------------------	-----------------------------	-----------

Files	17
Cables	20
Modems	30
Phone Lines	37
Summary	39

Chapter 4	A Tour of MicroPhone	41
------------------	-----------------------------	-----------

A Bit of History	42
FILE Menu	46
SETTINGS Menu	49
PHONE Menu	72
SCRIPTS Menu	75
FILE TRANSFER Menu	75

Chapter 5	Communication Command Languages	83
------------------	--	-----------

Time Invested vs. Degree of Automation	84
Low-Level Automation	88
Our Ideal Script Language	93
Discussion and Comparison	102
Summary	108

Chapter 6	Links and Hints	113
------------------	------------------------	------------

Macintosh to Macintosh	113
Macintosh to Macintosh Through an Intermediate System	119
Macintosh to Apple II Family	122
Macintosh to IBM PCs	123
Macintosh-to-Mainframe Services	131
Macintosh-to-Mainframes and Minis	133
Micro-to-Mainframe Software Solutions	138
Upward Mobility: The AppleTalk Network	141
Comparing Terminal Software	144

Chapter 7	Telephone Management	153
-----------	-----------------------------	------------

Call-Waiting Tip	155
On-Line Logs	157
Software for Telecom Management	163
The Business Communications Crisis	165

Chapter 8	Advanced Technical Topics	169
-----------	----------------------------------	------------

Files and File Formats	169
Using BinHex	175
Cables	178
Synchronous and Asynchronous Modems	188
Error-Checking Transfer Protocols	190
Data Security and Encryption	198
Standards Organizations	201

Epilog	TeleFutures	203
--------	--------------------	------------

Appendix A	FILE to FILE Import/Export Charts	207
------------	--	------------

Appendix B	ASCII and EBCDIC	217
------------	-------------------------	------------

The ASCII Character Set	217
EBCDIC	218

Appendix C	Sources Directory	221
------------	--------------------------	------------

Microcomputers	221
Communications Accessories	222
Macintosh Plus Cable Supplies	222
Phone Networks and Information Services	223
Names and Addresses of Manufacturers Mentioned in the Text	227
Directories of On-line Information Sources	233

	Work-at-Home and Freelance Consultant Information	239
	Government Information	235
	Additional Special Use Software	236
	Special Interest Networks (SPINs)	237
	Investor's Workshop	237
	International Association of Cryptologic Research	237
	NASA's BBS	237
Appendix D	A Brief Catalog of Micro and Terminal Families	241

	Macintosh Family	242
	PC DOS/MS DOS Family	242
	Apple II Family	246
	Terminal Families Chart	246

Appendix E	Two Softstrip® Data Strips	249
-------------------	-----------------------------------	------------

	BinHex 5.0	249
	DAFile	250

Appendix F	Modem Commands and Result Codes	253
-------------------	--	------------

	Glossary	259
--	-----------------	------------

	Bibliography	267
--	---------------------	------------

	Feedback Sheet	269
--	-----------------------	------------

	Index	271
--	--------------	------------

Figures

I.1: Information in Motion	xvii
2.1: Doin' the Info Shuffle	10
2.2: Questionnaire	12
3.1: Macintosh to IBM Flowchart	18
3.1: Macintosh File Forks . . .	19
3.3a: Macintosh to ImageWriter Cable	21
3.3b: Macintosh Plus to ImageWriter II Cable	21
3.4a: ImageWriter Cable	22
3.4b: ImageWriter II Cable	22
3.5: Macintosh Printer Port	23
3.6: Macintosh Plus Printer Port	23
3.7: Data Bus	25
3.8: RS-232 Connector	27
3.9: RS-422 Connector	28
3.10 Apple Direct-Connect Modem Photo	31
4.1: FILE Menu	47
4.2: Human and Computer Communications	48
4.3: Settings File Icons	49
4.4: SETTINGS Menu	50
4.5: Communication Settings	51
4.6: Prisoner's ASCII	51
4.7: Character Block of 10 Bits	58
4.8: Terminal Settings	60
4.9a: Full Duplex with Remote Echo	62
4.9b: Half Duplex or Full Duplex without Remote Echo	62
4.9c: Local Echo Only	63
4.9d: Local and Remote Echoes	63
4.10 File Transfer Settings	65
4.11: Flow Control	66
4.12: Startup Action	71
4.13: PHONE Menu	73
4.14: Set Up New Service	74
4.15: SCRIPTS Menu	75
4.16: FILE TRANSFER Menu	76
4.17: Macintosh File Forks	77
4.18: Word Save Dialog Box	77
4.19: File Transfer Settings Revisited	82
5.1: Lazy Admiral	85
5.2: Automation vs. Time	86
5.3: Scripts Compared	87
5.4: MicroPhone Directory	88
5.5: inTalk Macros	89

5.6: inTalk Buttons	91
5.7: inTalk MacBinary	93
5.8: Watch Me/Write Me . . . Make Me Do Fast Scripts	94
5.9: Smartcom Timing Control	95
5.10: inTalk Attack Dial	95
5.11: Smartcom Branching	96
5.12: Smartcom Protocols	97
5.13: MicroPhone Remark	97
5.14: MicroPhone Debugger	98
5.15: MicroPhone Script	100
5.16 inTalk Script	100
5.17: MicroPhone Application Generator	103
5.18: MicroPhone 'Watch Me'	103
5.19: inTalk Debugger	105
5.20: Red Ryder 9.4 Menu	107
5.21: Smartcom II's Graphic Box	109
5.22: Side-by-Side Comparison (MicroPhone v. InTalk)	109
6.1: Macintosh to Macintosh	114
6.2: Mac to Mac via Intermediate	119
6.3: Macintosh to Apple II	122
6.4: Macintosh to IBM PC	123
6.5: MacCharlie	128
6.6: MacCharlie Block Diagram	130
6.7: Macintosh-to-Mainframe Services	131
6.8: Macintosh to Mainframe	133
6.9: AppleTalk Network	141
6.10 Telecom Software Comparison Chart	147
7.1: Quick-Reference Sheet	161
7.2: MacMail/MacGeorge Transfer Log	164
7.3: Sidekick	164
7.4: Sidekick Log	165
8.1: Gibberish File	170
8.2: File Filter	171
8.3: Bit-Mapped Graphics	172
8.4: QuickDraw Graphics	173
8.5: Text	173
8.6: Cellular Info	174
8.7: Bundle Bit	174
8.8a: Smartmodem Cable	179
8.8b: Apple Modem Cable	180
8.8c: Macintosh Plus to Personal Modem/ImageWriter II Cable	180
8.8d: ImageWriter Cable	181
8.8e: Direct Connect Cable	182

8.8f: MacLink Cable	183
8.8g: Macintosh Plus Adapter Cable	184
8.9: Telebit Bandwidth	190
8.10: MacBinary Header	196

Tables

7.1: Approximate Times, in Minutes, for File Transfers	158
8.1: MacBinary Format Description	197

Checklists and Planning Aids

Telecom Questionnaire	12
Sample Program Design	98
Macintosh to Macintosh via intermediate system (steps)	120
Fixing a Wordstar File (steps)	127
Macintosh to DEC machine (steps)	135
Macintosh to IBM 3270 (steps)	136
Telecom Management	143
Phone Consultation	158
One-minute Telecom Manager	159
Soldering Tools	186
Soldering Hints	187
Security Precautions	201

Acknowledgments

Many individuals and companies helped us in researching *MacAccess*. Although the responsibility for its final written form rests with the authors, we would like to thank the following people in particular for their support during this project.

Etel Barborka
Matthew Bronson
Dennis Brothers
Kimman Chang
Frank Da Cruz
Dzintar Dravnieks
José Garcia
Greg Gerdy
Bill Grout
Scot Kamins
Jon Henry Kouba
Leo Laporte
Yves Lempereur
Wade Myers
Pamela Minet
Mike Quinn
Andres Sender
Neil Shapiro
Martha Steffen
Robert Triptow

Introduction

For those crucial times when you want to get something out of your Macintosh and into someone else's computer—and *quickly*—you'll want to send the information directly, computer-to-computer. The alternative of printing it out, mailing the document, and then rekeying the information into the second computer is time-consuming and expensive. *MacAccess* tells you how to make the right connections and keep your information in motion (Fig. 1.1).

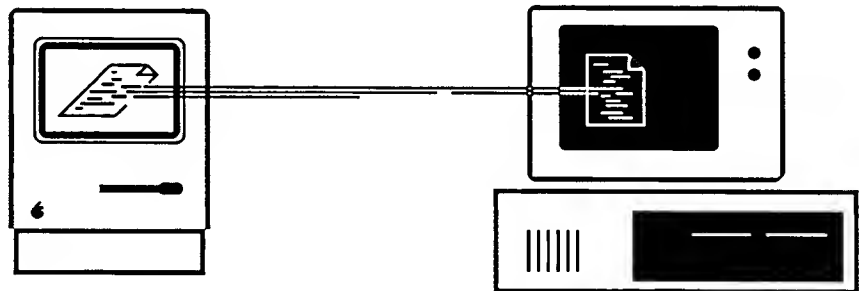


Figure I.1 Information in Motion

Your Macintosh has made the acquisition and creation of new information easier than ever. Until now, most books about telecommunication (telecom) and electronic networking have emphasized *acquiring* information from large, so-called information utilities. Throughout this book, though, we assume that you are an originator and creator of information in your own right. Your headwork adds value or expertise to the information you acquire from outside sources. Furthermore, we assume that a lot of the information you are interested in acquiring does not always reside in commercial on-line data bases, but more likely lives in another Macintosh, in a colleague's PC, or in your office mainframe. If it resides in a mainframe, the information may be more or less zealously watched over by managers whose cooperation you will have to gain in order to access what you need to accomplish your job.

This book will help you and your Macintosh talk to those managers and their mainframes. What's more, it will show you how to connect your micro to your colleague's micro in the absence of any heavy-duty, expensive Local Area Network (LAN) solution.

Note: We realize that some situations will require the expense and trouble of a *true* Local Area Network and not just a hard disk and printer-sharing system. Such systems are beyond the scope of *MacAccess*, although we do provide a few pointers in that direction in Chapter Six.

Finally we assume that once you've designed a report, written a newsletter, or created some charts and graphs, you won't want to stop there. Before the information you've created gets to paper—which most of it eventually does, contrary to the myth of the paperless office—you may have to send it to someone else for further processing or review. Information isn't any good unless it's moving. The day-to-day reality of today's information environment is that the majority of projects are worked on by more than one person; most projects have become too complex and too big to be handled by one person. Streamlining work and sharing information within and between work groups is what personal computers are all about.

If you're going Macintosh-to-Macintosh with your data, simply making a copy of a disk and sending the copy is a straightforward way to get information from here to there. If "there" is across town or across country, though, you have to cope with the inevitable delays and expenses that accompany sending physical objects via U.S. Mail or an

expensive courier service. Sending a disk is one step more convenient than sending paper, especially if the other person is going to have to do further processing of your information using the second Macintosh.

But what if the other person doesn't have a Macintosh? Or what if someone else, with either a different kind of software package or a different kind of micro, needs to get that information to you *now*, with delays of minutes rather than hours or days, in a form that will let you use it immediately?

That's where *MacAccess* comes in. We assume that you do not work in a vacuum. The information you need and the information you create are two aspects of the vital link between you and your clients, coworkers, or managers. This book will show you how to make the necessary connections quickly and easily, so that when you need to put information in motion, you'll know how to get your Macintosh to access and move it.

MacAccess is for all who use a Macintosh in their work. If you are a "gold-collar worker," you may have two or more computers at your workstation. They may both be Macs. Then again they may not. We'll show you how to link them easily and simply.

Note: According to Robert Kelley in *The Gold-Collar Worker* (Kelley 1985) gold-collar workers are those who "...work with their brains—not their backs." They use computers as extensions of their thinking to *leverage* their productive capacity.

MacAccess is for entrepreneurs, cottage industrialists, desktop media moguls, consultants, and specialists who work with Macintosh-based information. Armed with *MacAccess*, one-minute managers and members of corporate project teams will be able to move information with a minimum of administrative overhead. In short, *MacAccess* can assist you in connecting your Macintosh work environment to the rest of the world.

How This Book Is Organized

Questionnaires and Checklists

From time to time throughout *MacAccess*, we provide helpful questionnaires and checklists that can guide you through a particular procedure or technical area. The first one, presented at the end of

Chapter Two, will help you decide whether and how much you need to use telecom in your daily work. Others will assist you in comparison shopping for terminal software, figuring out the best way to move a file from one place to another, and deciding which applications software to consider when file exchange compatibility is an issue. There is a master list of these applications at the end of the Table of Contents.

MacAccess Tips

MacAccess Tips begin in this chapter and continue through later chapters. These are Macintosh-specific hints and tips you'll find useful in your own information transfers.

Tech Tips

Tech Tips begin in Chapter Three and can be found throughout the book thereafter. These are marked boxes containing supplementary technical information related to the sections in which they are found. Novices and those looking for specific information may safely skip these notes.

Chapter Summaries

Chapter One provides the basic discussions and definitions you'll need before proceeding with *MacAccess*. For example, terms such as *modem* and *information* are defined and introduced so that you have an overview before you get to the technical specifics that are covered in later chapters.

Chapter Two provides a global view of the process of linking up and transferring information. We present this view in the form of a short scenario involving two coworkers. Our scenario shows the process of information transfer from a hands-on perspective. At the end of Chapter Two is a questionnaire that can give you many ideas about how and when to use telecommunication.

Chapter Three digs into information transfer. We look at the physical parts of the system required to move files. We also examine the files themselves. You'll need this working knowledge in order to move information successfully. As you may have already surmised, there are many ways to keep information moving. We use the most typical setups in our examples. The example configuration in Chapter Three is *not* the least expensive, but it is one of the more flexible and powerful combinations of equipment and software. It has the added advantage of containing almost all of the components of just about any information transfer requirement you may encounter. Chapter Three

also covers the ins and outs of modem shopping and comparison. We provide a checklist of the features to consider when choosing modems. This chapter is one you should *not* skip, even if you already have some telecom experience. The rest of *MacAccess* assumes that you have acquired the major pointers presented in Chapter Three.

Chapter Four is a tour of a terminal program called MicroPhone for the Macintosh. During the tour, we cover more of the details of linkup in several self-contained discussions of many of the technical terms you'll encounter in data telecom, such as *start bits* and *baud*.

The final section of Chapter Four is a Macintosh terminal and communications program comparison chart. We include specific programs here, but you can generalize the chart to include other programs that you are interested in.

Chapter Five is about automating and customizing your communications so that more of the detail work can be taken care of behind the scenes. Making information transfer more automatic, whether over the phone lines or through direct cable connections, requires some knowledge of *macros* and *communication command languages*. The more sophisticated programs do not require you to be a full-fledged communications expert, or even a programmer, to take advantage of an automation language.

Chapter Six is a more detailed guide to telecom, covering specific machines. We show you the essentials of linking the Macintosh to other Macintoshes, to PCs, to Apple IIs, and so on. This is the place where you'll find our machine-specific advice and recommended hardware and software combinations. In this chapter, you only need to read about the specific configuration you're interested in. However, the hints and tips can help you approach the job of connecting two systems that we *don't* specifically mention with greater confidence of success.

Chapter Seven is about telephone management. If you do any significant amount of information transfer through your regular telephone lines, you'll want to manage your phone costs carefully to save money and remain competitive. Telephone management is an essential part of any business communication strategy. The Telecom Reference Sheet we provide will help you keep track of your on-line accounts.

Chapter Eight covers advanced technical topics encountered briefly in Chapters Two through Six, including such things as file conversion, cable wiring tips, and error checking. The sections on error checking and security will help you determine how much time, effort, and money you need to spend on system security and data integrity in your work.

The **Appendixes** contain additional reference material.

Appendix A is a table of import and export features of some of the more popular Macintosh programs. This table will help you determine whether or not a file has to be converted before it can be exchanged with another system running another application.

Appendix B is a set of ASCII charts. Such charts are helpful in troubleshooting communications problems, and they're a traditional part of books about telecom.

Appendix C is a directory of useful sources for additional tools, information, and techniques. This is a selection of items which we feel have the highest ratio of information value to cost, and which we also feel have relative longevity, given the rapid pace of change in telecommunications. Our emphasis is on "pointer" sources which are, themselves, resources that add value to the information, product, or service sold to you.

Appendix D groups some of the more popular microcomputers into related families according to the type of operating systems being used.

Appendix E makes BinHex 5.0 and DAFile available to you via Softstrips®. If you have a Cauzin Softstrip reader you can immediately load these utilities into your Macintosh.

Appendix F is a set of tables listing the commands and result codes for the Apple Personal Modem and Hayes Smartmodem 1200.

In the **Glossary** we include Macintosh-specific terms such as *dialog box* and *double-click* just in case you aren't a Macintosh owner and want to be able to communicate with those who are.

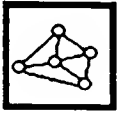
The **Bibliography** lists publications you may wish to consult for additional information and research.

Finally, a Feedback Sheet is provided to encourage you to communicate your experiences with *MacAccess* to us. Please be sure to read it.

MacAccess Icons



Throughout *MacAccess* this icon calls your attention to either background information or *MacAccess* Tips. Background information is part of the "lore" of telecom. It includes rules of thumb: technical specifics that need to be known by anyone trying to get the most out of their Macintosh communications. *MacAccess* Tips provide pointers not usually included in technical manuals.



This denotes networks of any kind: personal computers, a private network, anything having to do with networks in either the physical or metaphysical senses. (An example of the latter: kinship networks.) The telephone system is also a network, but it has its own icon. We also use this symbol to denote networks that are set up using existing phone network services, such as the regional Bell Operating Companies (BOCs) and services such as Sprint or MCI.



This icon denotes information about application software other than the specific packages we talk about in this book and whose icons are used in some instances.



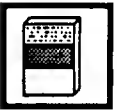
The Files icon introduces material directly related to the specifics of various kinds of files.



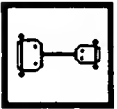
This icon identifies Macintosh-specific topics. In some of our schematic hookups, it also represents the physical Macintosh as connected into a communications link of some kind.



This icon stands for any non-Macintosh system: IBMs, Apple IIs, and so on.



This identifies mainframe-related topics.



This identifies anything about physical connectors, wiring, and signals thereof.



This denotes modem-related topics.



This icon identifies topics related to telephone services or phone lines.



This is used for technical information related to the general topics at hand. It is suitable for general consumption but is not necessary for successful communication.

CHAPTER

1

Coming to Terms

Do the headwork before the handwork.

Anonymous.

For the rest of *MacAccess* we'll be using terms that refer to physical things (your Macintosh, modems, and cables, for example) and words that refer to concepts and ideas (such as information and softspace). Terms that refer specifically to the Macintosh environment, such as icon, mouse, and scrapbook, will also be used, but we will assume you are already familiar with these. (We have included many of them in the Glossary, just in case.)

To prepare you for the rest of *MacAccess*, we present you with the following explanations. If you are already familiar with words such as *modem*, you can skip these explanations. If this is your first exposure to these words, we recommend that you take a few minutes to read these discussions.



Softspace

A *softspace* can be thought of as the whole system of which your Macintosh and you are parts. A softspace is “soft” because the definition of its boundaries changes depending on what you are doing.

When you connect with an outside computer, whether another Macintosh or a company mainframe, that other system and the physical connections between it and your Macintosh temporarily become part of your softspace. At different times, a softspace can encompass

of the following: phone lines, computers, computer peripherals, furniture, support materials (reference manuals, disk storage containers, and so on), networks, and other people. A softspace, then, is the partially conceptual and partially physical space in which you and your Macintosh play the central unifying roles in the system under consideration.



Files

A *file* is any block of information stored in electronic form on a floppy disk in your computer or some other electronic “holding tank.” In *MacAccess*, the name of the game is to get the file from a disk to a file on your Macintosh disk *in a form that you can use*, and vice versa. In some cases, you may have to perform more than one step to accomplish your objective. (When that’s necessary, we’ll show you how, step-by-step.)

Consequently, we can say that as far as this book and your work with the Macintosh are concerned, we’re *file-oriented*. The file can be anything: manuscript, spreadsheet, data base information, lists, computer programs, musical scores . . . anything! In non-technical terms, a file is any block of symbols that always has to stay together and move as a group to remain meaningful. When we talk about the *integrity* of a file or, more generally, about *data integrity*, we’re talking about the file’s internal order and reliability. A file that gets altered or changed unintentionally (such as during a transfer to another machine) is said to be contaminated. We have to be especially concerned about data integrity when information is in motion.



Modem

A *modem* is a piece of electronic equipment used to connect your Macintosh to the telephone network so that you can send and receive information that can be “understood” by computers. For the most part, we don’t care *how* a modem does its job. Chapter Three covers the essentials to keep in mind when you are choosing and using a modem, but we won’t be interested in details such as how a signal is converted from digital (computer) to analog (phone line) form. The modem should do its job and stay out of the way while we concentrate on our files.



Telephones and Phone Lines

There was a time when we wouldn't have had to define these terms. In those days, everybody had a phone and all the phones connected to the same thing and they all worked pretty much the same way. Today, the variety of telephones and telephone features is astonishing, and it is proliferating as the electronic universe itself continues to expand. Moreover, your phone company may not be my phone company. Worse, your electronic mailbox may not be on my network.

In all but the most exceptional of cases (which we will cover later), we can safely ignore the phone company and the phone lines entirely and get on with what we're trying to do with our files. In this book, *phone* and *phone line* refers to whatever commercial phone network you are using.

Like the modem, the phone company should stay out of the way, not seen and, preferably, not heard. The fact that it sometimes doesn't is the reason we still have Public Utility Commissions. As to the electronic mail incompatibilities, we'll just have to sort them out, the same way we've managed to sort out our telephone buying decisions. As time passes, sending electronic mail will become as easy as (yechh!) licking a stamp.



Software

Software is any structured information needed by computers to perform the tasks set for them by people. Contrary to modems, which do our bidding quietly and with no fuss, a given piece of software may give you a lot of grief along with performance, unless you know how to manage it or, more to the point, manage *with* it.

Although our primary concern is Macintosh terminal and communications software in this book, we'll be mentioning other kinds of software and specific application packages throughout. Many writers and companies have attempted to create classifications of software based partially on function. But in the real world, where the ultimate goal is complete flexibility and freedom with regard to the creation and manipulation of all kinds of information (text, graphics, sound, voice, data, and so on), the categories break down as individual pieces of software and individual functions are integrated and reintegrated.

Generally, when we speak of *applications* software, we're referring to packages that help us create different kinds of files (or documents) with different kinds of end results (for example, words on paper). These packages include word processing programs (Word, WordStar, and MacWrite), graphics programs (MacDraw and MacPaint), and spreadsheet programs (Excel and Crunch). An applications software package, then, is anything that helps us create files in easy, familiar, and comfortable ways.

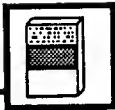
Telecommunications software is equally arbitrary in its definition: any software package whose main or primary function is manipulating existing files in order to move them or duplicate them in some other physical location separate from the hardware at hand.

Telecom software includes:

- *terminal* or *terminal emulation* software: any software that links your Macintosh with an Other Computer and, moreover, makes the Other Computer think that your Macintosh is one of its own.
- *micro host* software: any software that lets other computers or terminals dial into your Macintosh from outside and access selected files.

In general, telecom software is designed to work specifically with a modem or a direct cable link between your Macintosh and some other computer.

Utilities software can be very important in your day-to-day management overhead, especially as you begin communicating with more and more people. Any program designed to help you manage your computer itself, including its storehouse of files (your growing information gold mine) is a utility program.



Mainframes

The meaning of the term *mainframe* has blurred steadily over the last five years as machine capabilities have grown and physical sizes have shrunk. Usually, though, a mainframe computer weighs more than you can lift, costs more than you can afford, and is permanently stationed in a set-aside space, with or without security guards. We'll talk about particular mainframes and linking up to them in Chapter Six.



Micros

For us, the Macintosh is more than a microcomputer, so we use the term *micro* to refer generically to any other kind floating in the infosphere: PC clones, Apple IIs, TRS laptops, Amigas, and so on.



Networks

The definition of network depends on who's doing the talking and what his or her state of mind is at the time. (For a philosophical look at the network phenomenon, see Gengle's *The Netweaver's Sourcebook*).

We may sometimes refer to your *personal social network*, your two-machine, physically-hooked-together *micro network*, or your local *phone network*. There are also some more specialized meanings of network, as in the phrase *Value Added Network* or *VAN*. In all cases, the exact meaning of the term should be clear from the context in which network is used.



Information

The scientific definition of information does little to help us understand catchphrases such as *information is power* or *information age*.

Fortunately, there are ways of thinking about information that can help us manage our existence in the closing years of the 20th Century with much more panache, if not prosperity. When we talk about *information* in *MacAccess*, we'll usually be referring to some specific information contained in one of the files you want to move; or it could be information you've put in a letter or memo. For a wide-ranging discussion of the broader meanings and implications of the term, see *How to Win With Information or Lose Without It*, by Andrew P. Garvin and Hubert Bermont.



The following is a list of the names of the persons who have been appointed to the various committees of the Board of Directors of the United States Steel Corporation, for the year ending December 31, 1914.



The following is a list of the names of the persons who have been appointed to the various committees of the Board of Directors of the United States Steel Corporation, for the year ending December 31, 1914.



The following is a list of the names of the persons who have been appointed to the various committees of the Board of Directors of the United States Steel Corporation, for the year ending December 31, 1914.

CHAPTER

2

As the Modem Squeals

There is no time like the present.

Smollett

The benefits of well-managed telecommunication add up to that elusive, know-it-when-you-experience-it concept called “increased productivity.” You can increase accomplishment and decrease effort when you let the machines do more of the talking. To give you a better grasp of what we’re talking about, we’ve constructed a scenario for you. It involves two fictional co-workers. Only one of them uses a Macintosh.

Setting the Stage

An independent contractor hires herself out to businesses that need her skills as a researcher and troubleshooter; she’s a “mind for hire.” Her primary talents are:

- market research (competitor intelligence)
- project management (planning and budgeting)
- telepublishing (desktop publishing and telecom)

She networks in at least two ways. She uses telecom in her work, which means she uses phone and computer networks. She also networks social and business contacts who provide her with new job referrals and additional services that she sometimes subcontracts.

She works out of an office in her home and uses a Macintosh in her work.

Her roster of clients includes UpStart Software, Inc. UpStart Software develops generic computer programs that are sold at various retail outlets.

The other member of our scenario is our contractor's contact person and collaborator at UpStart Software. His official title is product manager.

As product manager, he has to coordinate production, sales, and marketing into a smooth, integrated process, turning out high quality products efficiently, reliably, and economically.

Just about everybody at UpStart software has an IBM PC on his or her desk.

The Scenario

Our independent contractor is in the midst of work on her current project, a market study for UpStart including assistance with their new product announcement and sales campaign. Suddenly, she receives an urgent phone call from Upstart's product manager. He is requesting the preliminary draft of the Zip Code demographics that she's been working on. He needs it right away for an important meeting.

Even before she hangs up the phone, our contractor has her Macintosh working on the emergency request. The information is in two separate Macintosh files. The first, an Excel spreadsheet file, contains the data and report formatting information that will be used in the final document. The second is a Word file that contains the text portion of the draft report.

She opens up her Excel spreadsheet and selects the relevant portions of the data, which she copies into the scrapbook. Then, she goes into Word and makes a copy of the text of the report. Using the copy, which will be a temporary document, she inserts the spreadsheet data charts one-by-one from the scrapbook into her report at pre-marked places. Finally, she saves the whole lot as a text-only file.

As soon as she has her new, integrated text-and-data file, she moves it to a special disk reserved specifically for telecommunicating. That helps her keep track of what has been sent to each client, when it was sent, and how much to bill for this part of her service. It also serves as a working disk for those times when she sets up her Macintosh to send information out overnight, automatically, in order to take advantage of the time she's not using her Macintosh and the lower long-distance rates.

Our contractor knows that the product manager will want to “tweak” her file (minor editing, formatting touch-ups, and so on) using his own word processor on his PC. The file she is sending, the file of origin or the source file, is what she calls “vanilla text” (formally, ASCII) which she knows can be used by many different kinds of programs running on many different kinds of microcomputer.

The product manager’s word processor is MultiMate, so when he gets the target file, he’ll do additional work with that program before getting the crash-produced report printed out in time for his surprise morning meeting.

Our contractor leaves her Apple Personal Modem connected to the Macintosh at all times. The modem, in turn, is connected to her regular telephone line.

She turns to the Macintosh Finder and double-clicks on the MicroPhone icon. She sets the Macintosh to its job of sending the file and turns her attention elsewhere. While she does whatever else she chooses, her Macintosh, the MicroPhone software package, the modem, and the telephone perform their chores in the background.

Under the control of the MicroPhone terminal software, the Macintosh dials up the product manager’s computer phone number at UpStart. That line has a PC (running a telecom software package for the IBM called Crosstalk) waiting for a call. The source file is almost instantly telecommunicated through the telephone wires and becomes a target file at the other end.

Our contractor frequently has to sell her clients on the benefits of telecom. Often, they don’t care about the technical details; they just want her to get on with the job. If she has to, she explains to her clients as much as they need to know to streamline the work. To be sure, she and the product manager had to do some preliminary work to get their micros to work together. The net savings in time and energy (messenger fees, photocopying, and on and on) more than pays back the upfront investment of time required to educate clients and set up the machinery to effectively share files electronically.

Now the product manager and our contractor routinely exchange all kinds of files with one another. Files created on the PC are regularly sent to her Macintosh for additional work. For example, she creates large numerical data files with Excel. She sends the data to the product manager for further processing using Paradox, a powerful relational data base for the PC family. Files he has created from Paradox, Lotus 1-2-3, and MultiMate are routinely sent back to her Macintosh.



Cartoon: Robert Triptow.

Figure 2.1 Doin' the Info Shuffle *

In the course of learning how to act as her own system integrator and system designer with regard to Macintosh communications, our contractor experienced a couple of unexpected side benefits: First, telecom-based processes have built-in advantages for time and project

*This image represents yet another way to create and share information. The cartoon was hand-drawn the usual way by the cartoonist. The drawing was then turned into a MacPaint bit-mapped image, using a scanner from Thunderscan on an ImageWriter printer. Once it's in a MacPaint file, the image can be sent over the wire to another Macintosh, where it can be printed out or used in other electronic documents. The image you see here is a reproduction of the scanned image as it looks when printed by a typesetting machine.

management. These include better cost accounting and, therefore, more accurate billing (based on closer tracking of actual costs) and better *throughput* (the rate at which action items get done).

Second, by building up her general knowledge of telecommunications and information handling, she has another area of expertise with which to attract clients, and one that can become a profit center in its own right.

Her bread-and-butter business depends on being able to have a variety of clients willing to continue paying for her expertise. Her ability to cater to their desire for instant information in a format that is more useful to them than mere paper-based information is what keeps them coming back to her.

Part of her strategic advantage in the marketplace is that she can provide the means by which she can work more closely with a client on a project. She and her clients become part of the same team much more effectively when they can work on the same electronic files together. She can't afford to turn away clients just because they don't have a Macintosh, and fortunately, she doesn't have to.

Is Telecom Worth It?

The skeptical old-timer from the mainframe data communications world may wonder whether the contractor really knows anything about stop bits, data forks, VT100s, parity bits, baud, bit rates, modulation and demodulation, file structures, error checking, null modems, cluster controllers, and all the other technical jargon associated with telecommunicating a file.

Far better questions for readers of *MacAccess* might be:

"How much of this do *I* really have to know?"

"Do I need to work this way in the first place?"

To answer the first question for you, recent advances in telecom software have made it possible for you to safely ignore most if not all of the technical infrastructure (or infostructure?). A lot of the things that you used to have to pay attention to in the "early days" of micros (1981 or '82) are now taken care of behind the scenes by the Macintosh and many of the software packages we recommend in this book. Even so, if you have a basic understanding of what's going on, which *MacAccess* provides, you'll be able to increase the power of the tools you have at hand.



Cartoon: Robert Triptow. Digitized with Thunder Scan.

Figure 2.2 Questionnaire

For the second question, we have devised the following non-scientific, “loaded” questionnaire to help you determine whether you can gain any real benefits from telecom. In ten minutes’ time, you’ll have a score that will help you decide whether you really need it or just think it’s a “good idea.”

Questionnaire

1. Do you have a Macintosh at home and use another brand of micro in your everyday work at an office located away from your home?
Yes [1] No [0]
2. Are you a self-employed person who uses a Macintosh as part of your daily professional life?
Yes [1] No [0]
3. Do you often need to find out a fact about something fast as part of your daily work?
Yes [1] No [0]
4. Would you like to find out more about your Macintosh and how it works without having to spend time in libraries and magazine stands trying to figure out which sources of information are the best for you?
Yes [1] No [0]
5. Would you like to find out more about how to operate a particular piece of Macintosh software without having to read another book, or at least without having to decide which book to read?
Yes [1] No [0]
6. Do you have a desire (overt or covert) to play computer games, design computer games, or find people to play computer games with?
Yes [1] No [0]
7. Would you like an easy, unique way to meet interesting people, attend "in" cocktail parties, and network with high-caliber people?
Yes [1] No [0]
8. Do you love tinkering with software (maybe you even secretly think of yourself as a "hacker" at heart) and enjoy discovering things, especially free software?
Yes [1] No [0]
9. Are you a careful consumer of microcomputer products and services who does a lot of comparison shopping and research before you buy?
Yes [1] No [0]

- 10.** Do you yearn for the day when you can do all your banking and shopping from home?
Yes [1] No [0]
- 11.** Do you want to be able to improve your professional standing through continuing education and lifetime learning programs?
Yes [1] No [0]
- 12.** Do you invest in stocks, bonds, and/or commodities (or want to) and want a better, easier way to manage your portfolio?
Yes [1] No [0]
- 13.** Do you dream of starting a home-based information business of your own someday?
Yes [1] No [0]
- 14.** Are you a sports fanatic?
Yes [1] No [0]
- 15.** Do you travel a lot and wish you had access to the same computer-based flight rate and schedule information that your travel agent uses?
Yes [1] No [0]
- 16.** Do you think of yourself as an “infomaniac”?
Yes [1] No [0]
- 17.** Do you need to track media stories that are printed or broadcast about your business, trade, or company?
Yes [1] No [0]
- 18.** Do you need to keep track of your competitors?
Yes [1] No [0]
- 19.** Do you frequently use Federal Express, Express Mail, or another courier service to send documents overnight or same-day?
Yes [1] No [0]
- 20.** Do you frequently send Telexes, TWXs, and/or telegrams to branch offices or a list of regular contacts?
Yes [1] No [0]
- 21.** Do you need to share information with someone else who has a different brand of computer than yours?
Yes [1] No [0]

- 22.** Do you have a need to keep in regular touch with a sales force, work group, or distant colleagues?
Yes [1] No [0]
- 23.** Do you have a product or service you want to market in new, more cost-effective ways?
Yes [1] No [0]

Scoring

- 0-5:** You're probably reading this book out of a vast intellectual curiosity. Perhaps if we expanded the questionnaire, which we could do since the potential applications are endless, you would find *something* that interests you and that would justify your purchase of a modem.
- 5-10:** You should buy a modem if you don't have one yet. If you have a modem, you know why you bought it and no further cost-justification is required. Now you just want to squeeze all you can out of your investment, which is probably why you are reading *MacAccess*.
- 10-15:** If you're not already using your Macintosh for regular telecommunicating, get ready for a quantum leap in your soft-space.
- 15-23:** You're under real pressure to produce results. You're into *MacAccess* and anything else you can get your hands on because you have immediate needs and you don't want to waste a lot of time before you can show some accomplishment.

CHAPTER

3

Digging into Telecom

Nothing is beautiful from every point of view.

Horace

The closer you look, the worse it gets.

Medical Student

The flow chart in Figure 3.1 is a graphic overview of one way to send a file from the Macintosh to an IBM PC. It is more-or-less similar to the process our contractor used in Chapter Two's scenario.

To fully appreciate the process, we're going to dissect it. We're going to dig in to find out more about just how a file of information becomes a message on the telephone wires, and how the message becomes a useable file again at the other end. Each of the sections that follow covers one of the major components of the communication process:

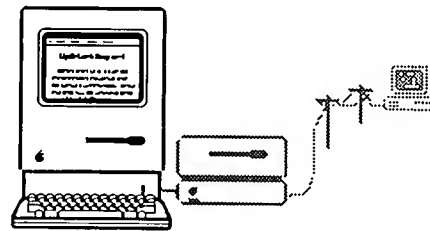
- Files
- Cables
- The RS-232/422 Standard
- Modems
- Phone Lines



Files

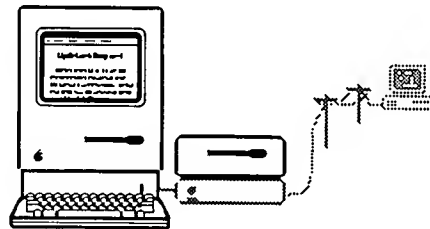
When you open a document (double-clicking its icon on the Finder desktop) to use, for example, MacWrite or Word, you are using a particular file, with its own name, stored on disk. Text documents,

spreadsheet data, programs, and MacPaint pictures are all stored in a similar way. All the information on the disk must be stored the way Macintosh (and the particular application) expects it to be stored, or it won't be useable. Other computers, using other application software, will use their own peculiar methods for storing their files.

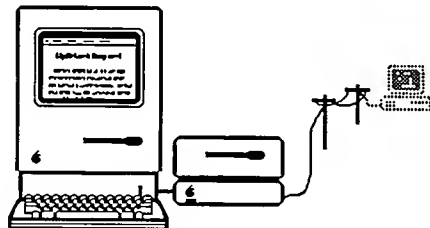


How to get a file from your Macintosh to an IBM PC

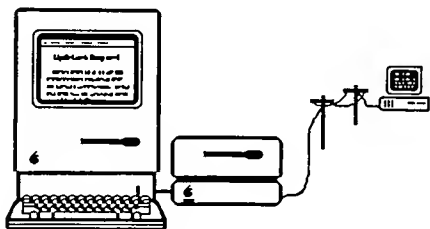
1. Create a File
 - Start application
 - Open document
 - Create or modify contents



2. Copy File to Transfer Disk



3. Connect Machines
 - Run telecom software
 - Open Settings file for destination
 - Dial remote system



4. Send File to Destination

Figure 3.1 Macintosh to IBM Flowchart

Note: There are no universally accepted, standardized ways of storing information on magnetic disks. There are certain standards followed by brands and families of brands. There are plenty of ways to bridge these *de facto* standards, however.

The predefined specification of how a file is to be laid out on the disk, what information is expected to be there, what the physical parameters are, and so on, are all part of the file's *format*.

Thus, the process of telecommunicating anything depends on meeting two requirements: getting the file (a bundle of related information) from here to there, and making sure the correct formatting information accompanies the file so that, once it gets there, it will still be useable. Of course, if both machines are Macintoshes, the file simply needs to get there with all its information intact and no intermediate transformations are necessary.

Which brings us to a fork in our narrative, or, rather, forks. Every file on your Macintosh disk can be thought of as a fork with two prongs, a data fork and resource fork. Another way of thinking about it is to consider every Macintosh file to be two sets of data that travel together and have the same name. Figure 3.2 is another, more visual way to think about the matter.

At any rate, no matter how you prefer to think about what's really going on here, the *data fork* of a Macintosh file contains just what its name implies—the data. Most of the contents of a Macintosh document (as opposed to an application) are stored in the data fork. It is the text of a letter, for example, or the numbers and formulas of a spreadsheet. When communicating with computers other than Macintoshes, this is the part you want to send.

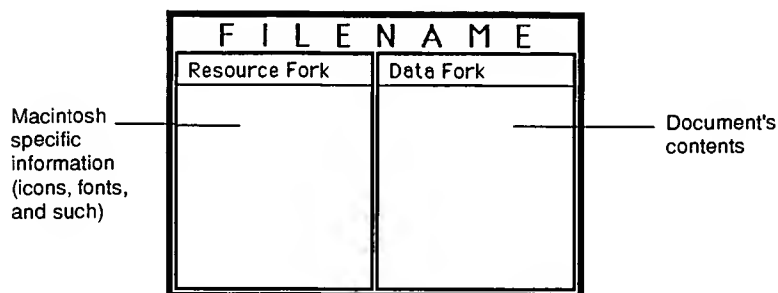
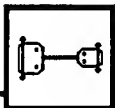


Figure 3.2 Macintosh File Forks...

The *resource fork* of a Macintosh file contains mostly stuff that's important or meaningful only to the Macintosh. For example, the resource fork holds the icon and font information of a file. Macintosh applications (as opposed to documents) are mostly stored in resource forks, and, in fact often have completely empty data forks.

When sending information to some non-Macintosh computer, you'll only be moving what's in the data fork, since the other computer won't be able to use the Macintosh-specific information. Before moving data between machines, it may be necessary to convert it to a different form. The conversion process is easy, for the most part, and you won't have to deal directly with the two forks—the Macintosh operating system keeps the fact of the split files hidden from users. Knowing about the data and resource forks, however, will help you understand the various file transfer protocols covered in more detail in Chapter Eight.

Many of the better Macintosh application programs now include menu selections that enable you to import and export files (see Appendix A). This makes it easy to get a file ready for transmission. Later we'll explore this idea of file conversion and transformation in greater detail.



Cables

We can't talk about cables without getting into more acronyms and numbers, but don't worry. These acronyms and numbers show up so often that they eventually become as familiar as your Social Security number.

Chances are, if you have a Macintosh, you also have an ImageWriter printer and, therefore, also an ImageWriter connector cable. If you have some other make of printer hooked to your Macintosh, our diagrams will help you figure out whether your particular printer cable is the same as the ImageWriter cable, and if not, whether and under what circumstances you can use it for other communications functions. Macintosh Plus users, take note: your lives are made easier by the design of the Mac Plus ports.

As it is, the ImageWriter cable happens to be what's known as a *null-modem* cable. The term has to do with the way the wires are paired between the connectors at each end of the cable. The null-modem cable crosses the wires on two of the pins, which gives rise to the other name by which null-modem cables and connectors are known: *cross-*

over cables or crossover connectors. (The wires that are “crossed-over” are the transmit and receive lines. In a null modem, transmit is connected to receive and receive is connected to transmit.) A null-modem cable can be used to connect an Apple II with the Mac, or the Macintosh to a letter quality printer such as the NEC Spinwriter.

Note: The ImageWriter cable *will not* work to direct-connect between the Macintosh and the IBM PC Serial Communications Adapter. See Chapter Eight for the recommended cable wiring.

Figure 3.3 is a schematic diagram of the cable (one for the Macintosh to ImageWriter—Figure 3.3a, the other for the Macintosh Plus to ImageWriter II—Figure 3.3b), and Figures 3.4a and 3.4b show a closeup

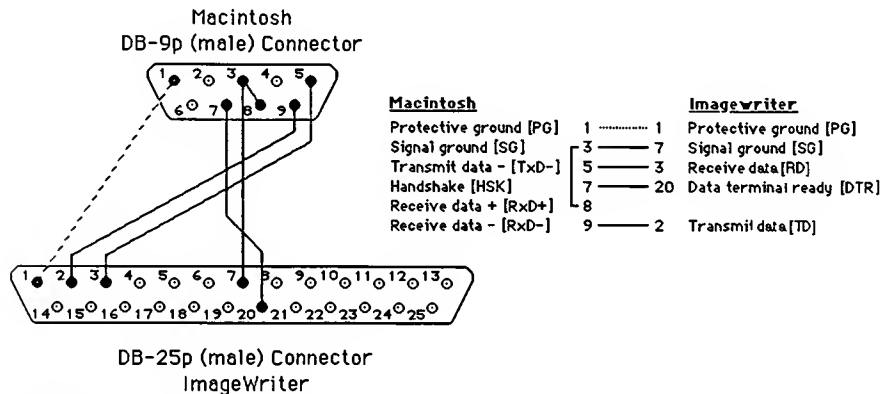


Figure 3.3a Macintosh to ImageWriter Cable

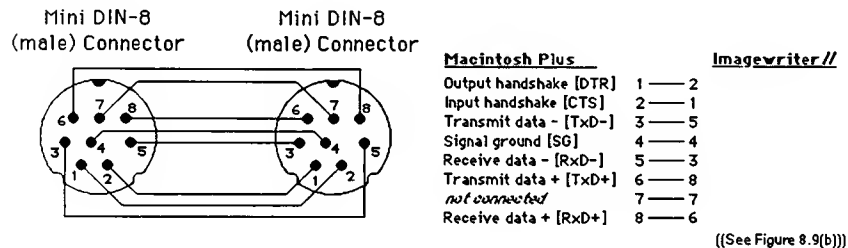


Figure 3.3b Macintosh Plus to ImageWriter II Cable

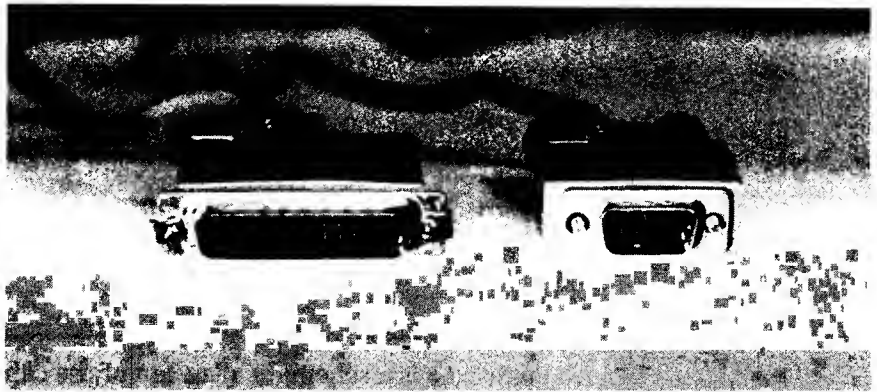


Figure 3.4a ImageWriter Cable

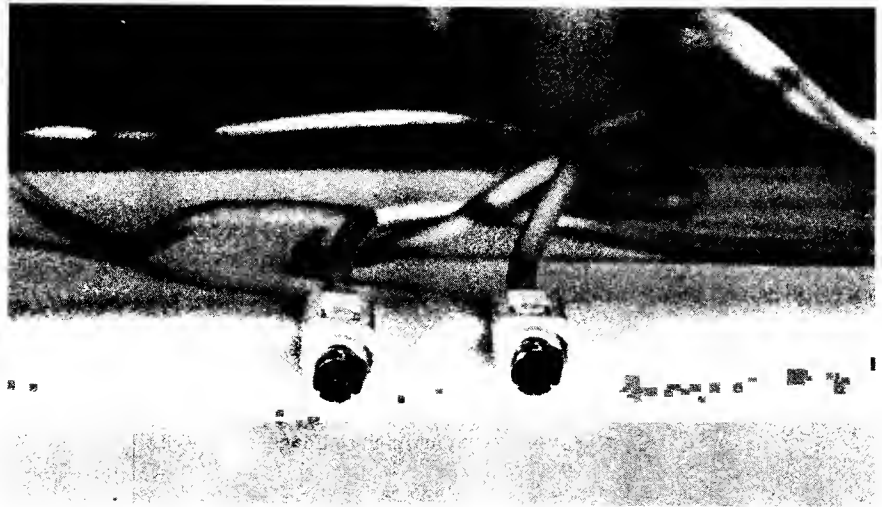




Figure 3.4b ImageWriter II Cable

of the two business ends of the cables. As you can see, cable wiring diagrams are not difficult to read. They are often included as part of the technical documentation that comes with modems or printers.

At the back of your Macintosh are two connectors marked with a printer icon  and a fanciful modem icon . These physical locations are also referred to as *ports* or *serial ports*. If you have a Macintosh, Figure 3.5 shows what the printer port connector looks like. If you have a Macintosh Plus, Figure 3.6 shows what your port connectors look like.

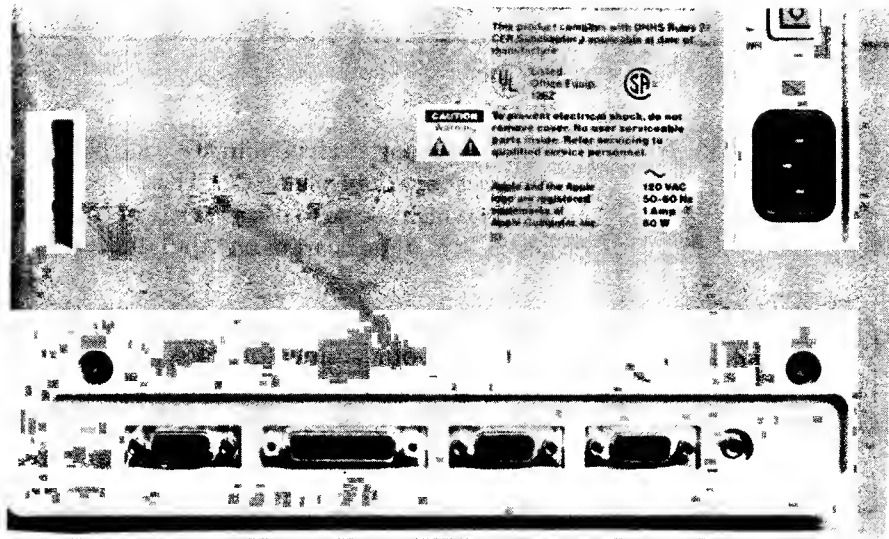


Figure 3.5 Macintosh Printer Port

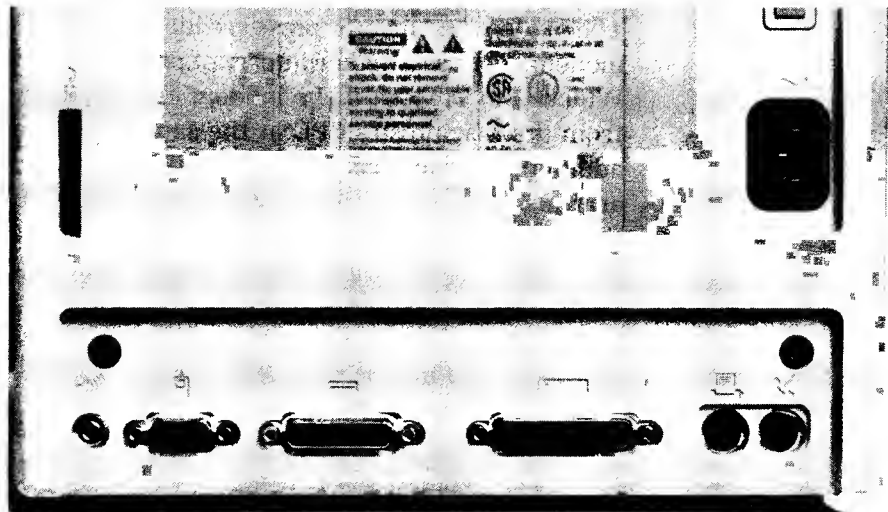


Figure 3.6 Macintosh Plus Printer Port



Tech Tip

Port Info

The two serial ports referred to are controlled by a Zilog Z8530 Serial Communications Controller (SCC). This is a powerful, complex chip in its own right which is able to use a variety of communication methods and protocols. It is at the heart of the AppleTalk low-cost LAN (Chapter Six).

Whatever your connectors look like on the outside, on the inside the Macintosh serial ports both conform to the Electronics Industry Association (EIA) RS-422 standard. This standard is an upgrading of an older one called RS-232. This makes the Macintosh pretty much state of the art where serial communications standards are concerned. The newer 422 specification is more resistant to line noise and interference and thus more reliable over longer distances.

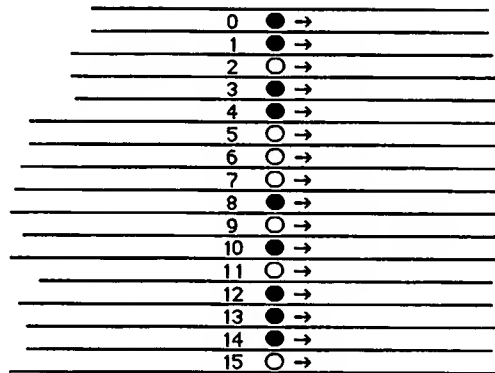
Unfortunately, much of the rest of the world, including the ubiquitous IBM PC/XT/AT crowd, still uses RS-232. Fortunately, RS-422 and RS-232 can talk to each other, because the 422 standard was designed to be compatible with 232.

Interrupt

The modem and printer ports are identical in all respects save one: the modem port has a higher interrupt priority than does the printer port. Which is to say that if the modem and the printer demand attention from the Macintosh's CPU at the same time, the modem will win. In addition, if your disk drive demands attention and characters are arriving at the printer port via either a direct cable or modem connection to another machine, you are likely to lose characters, since accessing the disk drives is also a higher priority item. If you want to use the printer port to communicate, either slow down the transmission ("bps" or "baud" setting) or set up an internal RAM disk to handle incoming data, thus eliminating disk drive access interrupts. Using this method, you can communicate successfully through the printer port at up to 9600 bps.

Parallel and Serial Communication

Inside most computers, information zips around in groups of electrical signals representing bits (0s and 1s). In the Macintosh, usually 16 bits move at a time (see Figure 3.7). The bits travel side-by-side, much like



The Macintosh data bus
transmits 16 bits at once

Figure 3.7 Data Bus

cars on a freeway. This is called *parallel* communication because the signals travel in parallel, with one wire allocated for each bit position. The advantage is that information is processed faster. The disadvantage is that many separate channels are required to carry the bits. More channels means more expense, especially if each channel is a discrete signal path of its own.

If we want to send all the data bits through a single wire, it has to be done one bit at a time. This is commonly called *serial* communication; bits are sent in a series, one after the other. One reason for using serial communication is to be able to send data bits over the telephone wires. The telephone system reaches into every nook and cranny of our extended softspace. It's downright *convenient* to use this existing medium (even though it is not the *ultimate* for computer-to-computer communication).

This complicates matters. Telephone lines don't have sixteen wires down which to send our groups of bits. Telephone connections only need two wires. So there has to be some way to translate the parallel communications that go on inside your computer to serial communications as required by external physical conditions. That's where RS-232/422 comes in.

Both the older RS-232 standard and the newer, compatible and more reliable RS-422 standard (used in the Macintosh) define a set of signals (and which wires they will travel on) for serial communications. There are still more than two wires used in the RS-422 and RS-232

standards (although you can sometimes get away with as few as three), but not as many as used in parallel communications. You can think of the serial interface standard, then, as a kind of intermediary between the parallel, multi-wire world inside computers and the serial, two-wire external world of telephone lines.

The RS-232 specification tells engineers and designers which signals (called, collectively, signal lines) will pass through a 25-pin connector. So there are two things we're looking at: what the signals are and on which wire or pin the specified signal will be placed. Even though the information exchange consists of serial bits, more than two lines are used: one each for sending and receiving data and various control and status signals (sometimes referred to as *handshaking signals*) are specified in the standard.



Tech Tip

What's a Signal?

A *signal* here means a *change in state*. Inside your Macintosh, electricity is what carries the signals. Most often, a signal takes the form of a change in electrical voltage from low to high or high to low. In the RS-232 specification, a change in electrical voltage may indicate that the computer is ready to receive bits from the modem if the voltage change appears on the Clear-to-Send (CTS) wire. A change in voltage on the data wires, on the other hand, signifies zeros and ones (bits). A modem changes the representation of bits (signals) from electrical voltage changes to sound (tones) that can travel on the telephone wires. The tones themselves, though, are also electrical on the phone lines. The only time you hear them is when you pick up an extension phone and listen to what the modem is doing. Of course, what's happening when you eavesdrop on your modem is that the phone receiver is changing the signals from electrical energy back to sound energy, which is what telephones were designed for. As far as information itself is concerned, a signal is a signal. We are only interested in whether or not all the bits come through. How many times the bits were carried by different kinds of signals is irrelevant, except that these changes or transformations might introduce errors.

Since your Macintosh and, usually, the other computer with which it is connected are both capable of sending and receiving data at

the same time, two lines are needed: one to send, one to receive. Simultaneous sending and receiving is called *full-duplex* communication.

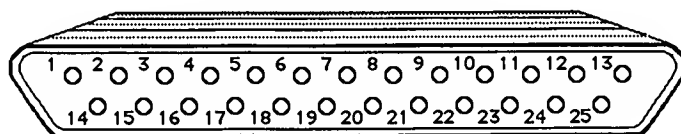
However, the original design of the RS-232 interface had to take into account that not all devices are capable of full-duplex operation. Devices that must take turns sending and receiving transmissions are called *half-duplex* devices. Half-duplex designs require that there be a set of control signals and a set of status signals that relay information about the state and activity of each device to the other. The control line for one machine is the status line for the other.

Furthermore, the originators of the standards decided to simplify matters by putting the computer side in control of the link between it and the modem (which in the early days was called a *data set*). The computer side is called the *DTE* (data terminal equipment) device. Modems are called *DCE* (data communication equipment) devices. Your Macintosh, then, is the DTE, and the modem connected to it is the DCE.

Figure 3.8 shows a diagram of the RS-232 connector with the control signals associated with each pin. Figure 3.9 shows the Macintosh RS-422 connector for the Macintosh and the Macintosh Plus.

The following definitions of each of the abbreviations help explain the common control signals and RS-232 lines.

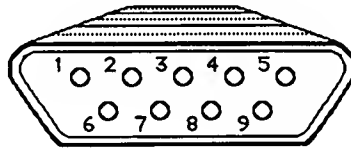
RS-232 Connector



1 Chassis ground (Gnd)	14 Secondary transmitted data
2 Transmit data (TD)	15 Transmission signal timing
3 Receive data (RD)	16 Secondary received data
4 Request to send (RTS)	17 Receiver signal timing
5 Clear to send (CTS)	18 <i>not connected</i>
6 Data set ready (DSR)	19 Secondary request to send
7 Signal ground (SG)	20 Data terminal ready (DTR)
8 Data carrier detect (DCD)	21 Signal quality detector
9 <i>not connected</i>	22 Ring indicator (RI)
10 <i>not connected</i>	23 Data rate selector
11 <i>not connected</i>	24 Transmit signal timing
12 Secondary carrier detector	25 <i>not connected</i>
13 Secondary clear to send	

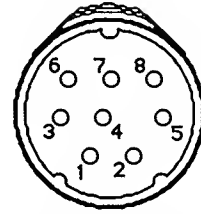
Figure 3.8 RS-232 Connector

Macintosh Connector



- 1 Protective ground (PG)
- 2 +5 volts
- 3 Signal ground (SG)
- 4 Transmit data + (TxD+)
- 5 Transmit data - (TxD-)
- 6 +12 volts
- 7 Handshake/external clock
- 8 Receive data + (RxD+)
- 9 Receive data - (RxD-)

Macintosh Plus Connector



- 1 Output handshake (DTR)
- 2 Input handshake (CTS) or TRxC
- 3 Transmit data - (TxD-)
- 4 Ground
- 5 Receive data - (RxD-)
- 6 Transmit data + (TxD+)
- 7 *not connected*
- 8 Receive data + (RxD+)

Figure 3.9 RS-422 Connector

- *DTR Data Terminal Ready* is generated by the DTE to signal that it is ready to receive data.
- *DSR Data Set Ready* is generated by the DCE to indicate that it is ready to accept control of the transmission. If DSR is not present, the device is either disconnected or in an invalid operating condition.
- *RTS Request to Send* is generated by the DTE to signal the beginning of the transmission. When using the half-duplex communications mode, this signal seizes the channel, blocking reception from the DCE.
- *CTS Clear to Send* is generated by the DCE in response to an RTS signal. CTS indicates that the transmission path is ready for data transmission.
- *CD Carrier Detect* is generated by the DCE to indicate that a carrier is being detected from the other side. Data may therefore be received. This is sometimes referred to as RLSD or *Receive Line Signal Detector*.
- *TD Transmit Data* is the line used by the DTE to send data to the DCE.
- *RD Receive Data* is the line used by the DCE to send information to the DTE.
- *RI Ring Indicator* is the line used by the DCE to indicate that there is an incoming call.
- *FG Frame (protective) Ground* is a connection between a modem chassis and a computer or terminal chassis to prevent damaging differences in voltage between them.
- *SG Signal Ground* is used as the reference conductor with which all other signal lines are compared.
- *TC and RC Transmit and Receive Clock* are generated by the DCE for synchronous transfers.

Note: We'll almost always be working with and talking about *asynchronous* serial communication, rather than *synchronous* serial communication. The latter is most often used with mainframes. For an explanation of the difference, see Chapter Eight, "Modems."

What needs to be remembered about the RS-232 interface standard is that the DTE end of the cable, the part that plugs into the computer, and the DCE end of the cable, the part that plugs into the modem, are not the same physically or electrically.

The RS-232 might better be called a "quasi-standard," because it has been implemented in different ways by different manufacturers. For example, when connecting two computers directly, without a modem, you can't use the cable that ordinarily connects your Macintosh to a non-Apple modem because both computers (two DTE devices) will try to control the communication flow through their RS-232 interface. Moreover, they will attempt to control the exchange using the same wire. To get around this, our special *null-modem cable* or *null-modem connection* is required. All the null-modem cable does is cross the TD (transmit data) and RD (receive data) wires. (More on this in Chapter Eight.)

Note: Apple Computer has altered the situation with its latest products (which, at the time of this writing, are the Macintosh Plus, Apple Personal Modem, and ImageWriter II). The same cable will connect a Macintosh Plus to another Macintosh Plus, a Macintosh Plus to the Personal Modem, and a Macintosh Plus to the ImageWriter II. Instead of using different cables to make the signals compatible between the machines, Apple switched the signals on the machines themselves. This certainly simplifies connecting this small world of equipment. It means, however, that special attention must be paid when connecting these machines to other equipment with more traditional signal configurations. See the cable diagrams in Chapter Eight for detailed information.

Which brings us back to the ImageWriter I cable used with the pre-Plus Macintosh. For all intents and purposes, this cable is a null-modem cable. When the situation calls for a direct connection between your pre-Plus Macintosh and another micro without using a modem, try this cable first and see if it works. In case it doesn't, we provide alternative wiring diagrams in Chapter Eight for making your own null-modem cables, along with a discussion about why you can often get

away with as few as three (namely, TD, RD, and SG) of the “quasi-standard” RS-232 lines connected.

Now you can see why we have spent so much space on cable wiring diagrams in *MacAccess*. It will be a long time before the standard gets straightened out; after all, a lot of the equipment we’re interested in connecting up with won’t be replaced with newer models for years. In the meantime, we’ll still be able to exchange information.

Seven Out of Twenty-five Pins Agree...

Sure, there are 25 pins on a connector at the end of that cable, but that’s probably 17 or 18 too many in any particular Macintosh connection. The main pins (signals) that are used are:

1. *Frame ground*
2. TD: Transmit Data
3. RD: Receive Data
4. *RTS: Request to Send*
5. *CTS: Clear to Send*
6. *DSR: Data Set Ready*
7. Signal Ground
8. *CD: Carrier Detect*
20. *DTR: Data Terminal Ready*

The pins shown in italics also may not be needed in some cases.

So why are they there if they aren’t used or aren’t needed? The full standard was meant to cover all possible conditions, and the standard assumes that control of the link will be performed by hardware rather than software. In many instances, effective connections can be made without using the full complement of hardwired signals by putting the link under software control. Fewer connections also mean lower cost.

Note that the Macintosh serial connections only have eight or nine pins.



Modems

Just about all your Macintosh telecom work will involve the use of a modem at both ends. In fact, even in cases where the two machines are located close enough to one another to connect using a null-modem cable, you might still want to communicate via modems and telephone lines.

In telecom, a modem is any device that *modulates* and *demodulates* a signal, changing it from digital to analog and back again. Digital signals are those used inside your computer, representing zeros and ones. Analog signals are those used on telephone lines.

Anything that puts information into a signal is called a *modulator*. Conversely, anything that extracts information from a signal and puts it into another form is called a *demodulator*. For example, when you speak into a microphone attached to a tape recorder, your voice *modulates* an electrical current inside the machine, which, in turn, changes the magnetic properties of specks of metal oxide on the tape. During playback, the tape recorder *demodulates* the signals on the tape, transforming them back into sound waves you can hear.

Internal and External Modems

The Macintosh requires an external modem, connected by a cable to the modem port, and by another cable to your telephone line. Other machines, such as the Apple IIe and the IBM XT, are able to use internal modems. These modems are in the form of circuit boards that plug into one of the empty slots inside the machine, rather than a separate box that must have its own cable. In practice, it doesn't make much difference whether a modem is an internal or external one.

Either way, most modems are *direct-connect* types as shown in Figure 3.10. They are connected electrically to the telephone wires through a cable that plugs into the wall phone jack on one side and the modem on the other.

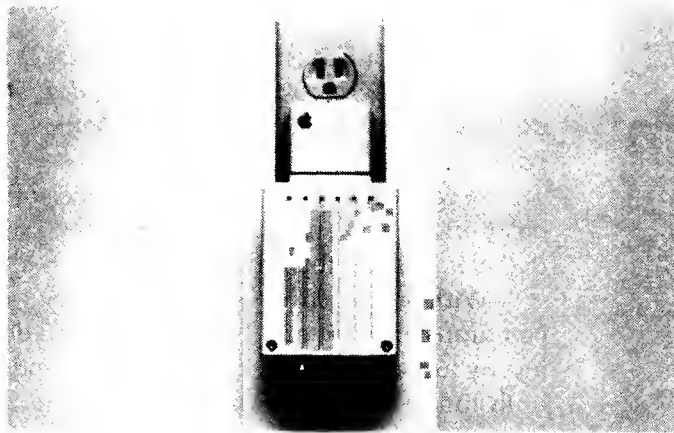


Figure 3.10 Faster than A Speeding Modem

In unusual situations, an *acoustically coupled* modem might be required. This kind of modem uses a speaker and microphone to inject its tones into the telephone handset and to “listen” to the other modem’s signal. No electrical connection is made to the telephone lines. Signals are coupled acoustically between the modem and the telephone network. The main disadvantage to this kind of modem is that it is subject to any noise that may be in its environment, which can blur or garble the signal. (See Chapter Eight, “Error-Checking Transfer Protocols,” for discussion of phone lines and noise.)

We recommend the use of a direct-connect rather than an acoustically coupled modem for the Macintosh. We tell you about acoustically coupled modems because you might, at some point in the future, be trying to connect with someone who is using an acoustically coupled modem. Any connection problems or noise problems would most likely be caused by the modem.

Modem Speed

Modems can be rated according to their speed. Generally, the faster the modem, the more it will cost. The most common speeds in the world of Macintosh communications are 300 and 1200 baud. Strictly speaking, 1200 baud modems are not really 1200 baud at all, but 1200 *bit-per-second* modems. However, in practice the *baud* label is often used to refer to speed, rather than bps, no matter what the speed is or how it is obtained. Throughout this book we use bps rather than baud. You can consider them to be interchangeable terms, since the rest of the telecom world treats them as synonyms most of the time.

Of Bits and Baud and bps

In telecommunications, *baud* is actually a unit of scientific measurement that refers to *the number of discrete signal states that can be detected each second on a given channel or connection between two points*.

Recall that a bit is the smallest piece of information that can exist: on or off, there or not there, zero or one, true or false. Most everything digital is built on this smallest unit.

With a couple of wires, a battery, and a light bulb, you can communicate a lot of information (if you don’t care how long it takes) just by turning the light bulb on and off and having someone watch the bulb and record the results.

We'll let one second with the light bulb on be a 1. We'll also say that one second with the light bulb off will represent a 0. Two seconds off is 00, three seconds on is 111, and so on. With this scheme you can represent all kinds of things, such as letters and numbers. There are two factors at work here which are of interest in understanding the difference between *baud* and *bits per second* as two different measures.

First, the number of discrete signal states (on or off, current flowing or not) communicated each second is one, and that signal state exactly represents one bit. The bit-per-second rate and the baud measurement are equal: one bit-per-second and one baud.

Let's upgrade our light bulb example. Now we'll use two telephone wires that together can carry four separate tones but only one tone each second. That's about as fast as we'll be able to identify and write down the tones as they're sent.

We'll say that each separate tone will represent one of four two-bit combinations: 11, 10, 00, 01. Further, we'll count them out into groups of four sets:

11100001

This set of eight bits just happens to be one *byte* of computer information.

By using this tone-to-bit and bit-to-tone modulating scheme, we're still sending signals at the rate of one baud. However, our bit-per-second rate has gone up to two.

In actual practice, this kind of signal-encoding scheme is what enables 1200-bps modems to work. With 1200-bps modems, there are still two discrete tones, but each tone has two distinct *signal phases* that can be detected with the appropriate circuitry. So instead of only two possible signal states (representing zeros and ones) we have four, one for each group of two bits, resulting in an effective fourfold increase in our throughput of bits.

In mathematical terms, *baud* is defined as the reciprocal of the duration, in seconds, of the smallest unit interval (that is, of the shortest signal element that can be detected and interpreted as a unit of information). In Morse code, for example, if a dot (the smallest unit in Morse code) can be sent in 10 msec (10 thousandths of a second) then the baud number (modulation rate) is 100 (1000/10).

Thus, it is only under certain conditions that the baud and the bps numbers will be the same. The bps rate will always be higher than or equal to, but never less than, the baud number.

There are still many systems that communicate at only 300 bps. The important thing to remember is that modem speeds must match. If your Macintosh is set to operate its attached modem at 1200 bps, then the modem (and the computer at the other end, as well as *its* modem) must also be able to communicate at 1200. If the other machine only communicates at 300 bps, then you must set the Macintosh to match. There are some 2400 bps and higher modems available, but because there is still a large installed base of lower-speed modems being used, most manufacturers of high-speed modems have made sure that they are “downwardly compatible”; that is, that the modems can communicate with 1200, 300, and 110 bps ones.

A 2400 bps modem is fast, but do you really need one?

There are some drawbacks. Like a high-performance sports car, 2400 bps and faster modems are still relatively expensive, although they continue to drop in price. The speed increase requires that these faster modems use some kind of built-in error-checking, due to the fact that the higher speed makes them more vulnerable to phone line noise. When an error is detected, data must be retransmitted and rechecked for new errors. The time it takes to do this is added to the total connect time, of course. When you add error-checking to transmission times, a 2400 bps modem may turn out to be as slow as a 1200 bps modem under some adverse (noisy) telephone conditions.

Many commercial services and data communications networks charge a premium for 2400 bps services. This means that although you'll be on line for less time, you're charged more per minute at the higher speed than you would be charged at 1200 bps or lower speeds.

The net result is that even if you deal with large documents all the time, the faster speed may not *necessarily* reduce either your actual connect times or your connection charges.

In addition, a given 2400 bps modem may not work with *all other* 2400 bps modems. Their modulation techniques (how they pack the bits into the signals) may differ. Worse, their error-checking protocols may differ. There are no standards. Or, once again in the electronics industry, there are competing standards. This inability of particular 2400 bps modems to communicate extends to many mainframes, most of which won't handle 2400 bps.

The conclusion from all the foregoing is that unless your needs are highly specific and well thought out, your best modem speed choice will probably remain 1200 bps. That will likely remain true through the

eighties and early nineties. In many instances, it will make more sense to wait and jump from a 1200 bps modem to a 9600 bps mass transit modem.

Eventually, we will all need these faster modems so we can send high-quality video images to each other along with our Macintosh spreadsheets and reports. We can already use existing 18,000 bps (that's 1,800 *characters* per second!) to communicate over ordinary phone lines (see Chapter Four). That's bound to catch on eventually.

A Rule of Thumb for Modem Shopping

A rule of thumb is that if you are going to be telecommunicating for less than six hours a week and if you won't be transmitting large files (more than one or two typewritten pages or one MacPaint document at a time) then you won't mind a lower-speed modem at all. However, chances are that eventually you'll want more speed, no matter what speed you begin with. So you might as well bite the bullet and get the best 1200 bps modem you can find. You can always find one more thing you want to do with it.

Here is a modem feature checklist. If you haven't selected your modem yet, this list will help you figure out what you need as opposed to what sounds good.

Price Shop around. Between manufacturer competition, technical advances, and dealer competition, bargain modems are easy to find.

Manufacturer Despite the bargains, you may want to consider the number of years a particular manufacturer has been in business and the number of years it has been making modems. The devices are pretty reliable, but they're still expensive enough to have to think about what you'll do if the modem breaks down. Is the maker still there to help? And what about customer support if you need help installing the device or getting it to work properly? Is it cheap enough to justify throwing it away if it breaks?

Speeds handled For the utmost flexibility, make sure that the modem can handle a variety of speeds from 110 to 1200. Most Hayes-and-Hayes-compatible modems can support intermediate speeds, too, provided that your terminal software also supports intermediate, non-standard speeds.

Auto-answer/auto-dial Don't even consider a modem that can't auto-dial using either tones or pulses (preferably both). In the long run, auto-answer is also a necessity.

Error-checking block transfer protocols You need to consider this factor only with the newer 2400 bps modems. 1200 bps and slower modems generally do not have error-checking protocols built into them. Such things are handled by the terminal software.

Built-in “smarts” and extra features These features include ringback, passwords, automatic transfer control, and built-in file buffers. Some “ultra smart” modems have enough power built in that you do not need to have your Macintosh control them. Rather, you can send a file of data to the modem’s buffer and instruct the modem to handle the rest of the communication transaction (dialing, logging on, sending the file, etc.) on its own. “Ringback” and password protection can also be built into modems. Ringback means that the caller dials in and gives a password, at which point the modem hangs up and dials the caller’s number and then connects. This is a form of protection against unauthorized callers.

Software support Does the modem come with a terminal program and/or is it supported by other terminal software publishers? Or does it just come with a printed manual telling you how to issue commands from the keyboard?

Command set supported The Hayes command set is commonly used with many terminal packages so that the software and modem work together with little or no attention from you. If you want to use a particular piece of software with a particular non-Hayes modem, then you may run into problems. For example, if you like and want to use Hayes’ Smartcom terminal program (see Chapters Five and Six), then you have to use a real Hayes 1200 bps modem or a modem that incorporates the full command set. Most terminal software will let you change its settings to work with the command set of your particular modem. Many modem manufacturers now include or make available some kind of terminal package designed to work with their specific modem type. Check these things out before you buy, however, especially if you are shopping for your modem and your terminal software at the same time.

The Hayes Command Set

Some time ago in Atlanta, Dennis C. Hayes began to produce a plug-in board and connector-box modem for the Apple II family of microcomputers. Subsequently, the MicroModem II, as it was called, became the best-selling microcomputer modem. It was low-cost, reliable, and connected directly to the telephone lines. It was “only” 300 bps but it worked, and it was partially responsible for introducing thousands of people to the concept of an “on-line community.”

D.C. Hayes & Associates became Hayes Microcomputer Products, Inc., and Hayes now markets a line of software and modems for IBM PCs, Apples, Macintoshes, and other computers.

Their modems all use what has come to be called the Hayes command set. This is a specification for the signals used by the modem to carry out the instructions of the computer (or more precisely, the terminal or communications software that is controlling the computer). For example, the string AT gets the modem's attention and tells it to get ready to accept a command. DT tells it to dial a phone number using tone signals, and so on. The Hayes command set and the Apple modem command set are compared in Table F1, Appendix F. The Apple implementation is considered to be fully compatible, because the places where it differs are considered upwardly compatible with the Hayes set.

Many other modem manufacturers have incorporated some or all of these commands in their own modem designs. Many communications software publishers have incorporated the commands in their terminal packages, which means that these software packages should all work the same with any modem that uses the Hayes command set, not just those from Hayes.

In practice, you may have to look closely at the issue of Hayes compatibility. Not all modem manufacturers who advertise themselves as Hayes-compatible actually implement enough of the Hayes command set in their modems to make them work with all software that also supports the Hayes commands.

For example, a modem may only respond to ATDT (dial the phone) and ATZ (hang up the phone by returning the modem to its default state) and little else. Whether this matters to you or not may depend on whether you've settled on a modem/software combination for your Macintosh yet. It may also come into play if you are in the market for a faster modem than the one you have now. If Hayes compatibility is important to you, make sure that the modem you're buying, if it's not a Hayes, supports a complete set of Hayes commands ("complete" as defined by what your software will expect to be able to use). It may also make a difference to you when you are trying to communicate with someone who is using a Hayes or Hayes-compatible modem, especially if it's a 2400 bps box. If you know their command set, you may be able to solve related communication problems more easily.



Phone Lines

Think about it for a moment: our phone system is at once marvelous and frustrating, a source of satisfaction and occasional irritation. Technically, it's supposed to just stay out of the way and not draw attention

to itself. There are times when it doesn't. The phone system used to mean just AT&T. It is no longer a system but many systems, and these systems are, in the U.S. at least, businesses trying to sell you something called "telecommunication services for the information age." They include such illustrious initials as IBM, GTE, and GE, and a lot more are coming.

The phone systems haven't been just wires or just analog communication for a long time. When we say "phone system," we're talking about satellite links and fiber-optic links. We're talking about microwave relays, all-digital PBXs, packet-switched networks, and so on.

We are forced, in today's universe of telecom possibilities, to look beyond our monthly phone bills at the infrastructure that helps generate them. Most of the problems stem from two basic factors: noise and distance. As one increases, so does the other, for various physical reasons beyond our discourse here. Noise on a voice phone line usually isn't that bad. If it is, you can ask your caller to hang up and you can try your call again and hope for a better connection. But if the signal carries digital information, noise can be a very bad problem.

If you throw enough money at any technical problem, it can probably be solved. Enormous sums of money are currently being invested by various players in the telecommunication carrier game to create an Integrated Services Digital Network or ISDN. Eventually, the result of this investment will be that the effect of noise on long distance communication of any kind will be close to none. On the other hand, it will probably cost us, the consumers, a lot more money than today's phone services.

Today, for example, when you want to send signals via satellite, the other of our two major factors comes into play: the long distances involved (46,000+ miles) create delays that can hamper high-speed computer communications. Even with a perfectly clear signal, the long distance itself is a basic technical problem for data communications.

For local communication, of course, the ordinary methods we cover in this book will suffice for accurate and speedy information exchange. The phone lines in any given area are quite robust. For most telecom work, they are also quiet enough. Still, a lot of the design of modems and communications software has to do with the fact that even on relatively good telephone lines, an occasional burst of noise will wipe out some of your bit stream and wreak havoc on your data.

Summary

This chapter addressed the question of how you get a Macintosh file—a bundle of 0s and 1s that must travel together—from the Macintosh to another machine and keep the information intact.

The first step is to understand the parts of a Macintosh file: the data fork and the resource fork. The data fork usually contains text information. The resource fork contains Mac-specific information, such as icons or Macintosh programs.

The next crucial link in the communication chain, and the one that most often gives even experienced technicians problems, is cables. A null-modem or crossover cable is specially wired to directly connect two computers. The RS-232 and RS-422 quasi-standards help define the way a cable is wired up; that is, they specify which signals are supposed to appear on which pins.

The cable, in turn, may be connected to a modem, a device for modulating and demodulating digital information.

Finally, the telephone communication system serves as the last part of the link to some other computer. Two major factors affect telecom with micros: distances and line-noise.

Those are the primary pieces of the connection. Astute readers may note that we haven't discussed communications software yet. Software *per se* is a wide-ranging topic. Needless to say, none of the rest of the parts we covered so far will work without software. In Chapter Four, we'll take apart one of the top Macintosh communications software packages on the market. Using that specific example as our starting point, we'll return from time to time to the various parts of the system described already.

CHAPTER

4

A Tour of MicroPhone

Everything yields to diligence.

Antiphones

You must know it before you can control it.

Anonymous

In this chapter, we'll take a software tour through a general-purpose terminal package called MicroPhone. More than a review, our commentary during the tour will include all the important nooks and crannies of telecom. MicroPhone incorporates all the features needed for everyday communications with your Macintosh. We use MicroPhone as a specific example of the necessary functions to look for in any communications software package. When we talk about how MicroPhone handles files, it serves as a way to talk about the ins and outs of proper file handling. We'll also take a few side excursions into helpful background material.

Our tour in this chapter builds on information presented in Chapters One and Three. If you haven't done so already, take some time to make sure you are acquainted with files, modems, phone lines, and the telecom basics presented there before continuing.

We picked MicroPhone for our tour because it is highly versatile and takes advantage of the Macintosh user interface, a boon to novice and technician alike. During our tests we found that MicroPhone performed well under a variety of typical telecom conditions.

Remember that every program has its own specific strengths and weaknesses. You've probably developed your own wish and hit lists for your favorite programs. Realizing this, a cautious person might decide

to acquire one of the public domain or shareware terminal programs (such as Red Ryder or FreeTerm; see comparison chart at the end of Chapter Six) and use that for awhile. That gives you direct experience with communications software at a much lower cost. Then you can refine your list of desired features before plunking down money on a product. Our tour of MicroPhone will assist you in deciding which way to go and will give you a good start at creating a “Wants and Features” list.

Keep in mind too, as we take our tour, that the whole field changes rapidly, especially software. The snapshot we provide here will give way to a new picture tomorrow. MicroPhone itself is undergoing revisions even as this is being written. If it remains successful and finds a wide enough audience of users, this process of continual evolution will continue.

The comparison chart at the end of Chapter Six provides a comprehensive overview of the features to pay attention to in today's communication software. With any consumer product, the features you will want or need will depend on your situation. Videocassette recorders and cameras, to use a familiar example, differ in feature combinations and price. In the same way, terminal software designers and publishers differentiate their products by adding features or by specializing in a “niche” market (which usually means performing some highly specific function with a high degree of automation). The basic functions, though, remain pretty much the same.

A Bit of History

The Macintosh was released in January of 1984. Dennis Brothers, a programmer and designer of terminals, was among those who bought a Macintosh within days of its release. He wanted to use his Macintosh to connect to other computers. He has had about 20 years' worth of telecom experience, and it isn't surprising that he didn't let the Macintosh's early lack of software or documentation stop him from using the Macintosh for telecommunicating. Given his background and experience, and the fact that he finds his work as a telecom professional spilling over into a hobby, he decided to jump right in and reverse-engineer the Macintosh's RS-422 interface and synchronous controller in his spare time.

It wasn't long after that (February 1984) that the first version of MacTEP (Macintosh Terminal Emulation Program) was available. This *shareware* product (freely distributed by the author, but copyrighted material) was available in some places before Apple's own MacTerminal could be found.

Time passed; MacTEP went through several revisions. Commercial terminal programs began to show on the Macintosh scene by mid-1985. Meanwhile, several additional shareware and freeware terminal programs began popping up on electronic bulletin boards and commercial services.

Somehow, in spite of the flurry of general-purpose telecom programs that were published through mid-1985, none of them really covered the spectrum of user needs in terms of both power and ease of use. The latter, of course, is what Macintosh is supposed to be about. (Often, designers leave out the former on the assumption that a product must be *either* for power users *or* for novices, but never for both.)

Many of the public domain programs are powerful pieces of work, but many lack user-kindness and offer little companionship for the stranger to telecom. These programs give up their rewards slowly and only after much patience.

Then there was MicroPhone. Released in the first quarter of 1986, MicroPhone is a robust, full-featured telecom product. Its lineage, as you might have guessed, is traceable to Dennis Brothers, who wrote MicroPhone as a result of his experience with MacTEP.

MacTEP is still available on many systems and in user-group software libraries, but Brothers no longer supports it or updates it. In fact, he claims that there are better freeware terminals out there now. He doesn't recommend them either, though. Instead, Brothers is devoting his energy to MicroPhone.

Freeware, Shareware, and Public Domain Software

People have asked us, "What is the difference between *public domain software*, *freeware*, *shareware*, and *vaporshareware*?" For an explanation, we have to turn to history.

First there was *public domain software*. When any creative work that is theoretically "copyrightable" is published without a copyright notice, that work is said to be in the public domain. Works can enter the

public domain when their legitimate copyright expires. Many of the classics of literature, for example, can be copied and quoted and republished by anyone because their copyrights have expired.

Software as a copyrightable genre doesn't have any "classics" that are old enough for much of it to have entered the public domain due to expired copyrights. Most of the software that is in the public domain is software that was published without a copyright in the first place.

Many programmers come from the academic tradition of freely sharing the information that results from scientific research. Besides, so much good basic programming has gone on that it's really foolish to have to "reinvent the wheel" every time a new application has to be written. Many programmers have placed their implementations of particular algorithms, utilities, and even whole applications into the public domain. That is, they published or otherwise made available their programs without any copyright notice or claim. The Xmodem communications protocol (see Chapter Eight) is a good example of this practice. Xmodem was begun and published by Ward Christensen, who made it available to anyone who wanted to use it. This was a good way to get it spread around and contributed to its continual development and acceptance. Xmodem is an example of a standard that took hold because there were few barriers, economic or legal, to its adoption.

As more and more programmers became entrepreneurs, however, the basic differences between the academic environment and the mass-marketing environment became important personal issues for many: the need to stake a claim to their intellectual work in order to have it support them conflicted with the tradition of openly sharing programming information and programs with one another.

Freeware was used to describe public domain software that was placed on a dial-up telecom system for distribution to anyone who wished to copy it. Then some programmers began to copyright their work, yet make it freely available to those who wanted to use it. They called their programs freeware, and they placed restrictions (legally untested) on its commercial use.

When the IBM PC arrived on the scene, at least one enterprising programmer used the term *freeware* as a proprietary company name. He placed his copyright notice on the software he wrote, along with a request that people send a small fee to him if they used the program and liked it. About the same time, other programmers began calling their software *shareware*, and also asked for a fee from users.

Now the use of these terms has become so fuzzy that you may find freeware, public domain, and shareware describing any of these cases:

1. The program bears no copyright. There may or may not be an author's name on the software. No fee is expected or even hinted at by the author. (There may sometimes be a fee for these kinds of programs which the distributor may ask as a "materials handling and shipping" fee. This is the case with many clubs and user groups that distribute freeware on disk.) The copyright law says you can use these programs any way you want: you can decompile them, change them, put your own name on them, combine them, give them away, or charge for them.
2. The program bears a copyright in the proper format, which means that the author's or owner's name (if a company) appears in the copyright line, but no fee is expected or asked. The program is made available at the owner's discretion to anyone who asks for it. Those who get copies may, themselves, copy it and give it away. However, usually some sort of notice accompanies the software to the effect that no subsequent acquirer of the software may sell it, incorporate it into another package which is then sold, or otherwise "profit commercially" from the product. The legality of such notices is questionable under current copyright practices.
3. The software bears a legitimate copyright and a notice to the effect that those who use the program are asked to send a fee of some kind to the author, usually less than \$50. This is sometimes called a "registration fee." In return for their money, users can freely use the software and are encouraged to copy the program and pass it on to others. The author or owner usually adds incentive to registration of copies by offering telephone support or update notices to registered users.
4. The software used to bear a legitimate copyright, but someone has removed the copyright notice and has given the program away or, worse, placed it on a system where it can be freely copied by anyone who dials in. Generally, the owner of the system(s) where such software can be found runs more risk than those who copy the software (whether they are aware of the program's origins or not) and use it. However, software publishers, through several trade organizations, are creating legal heat for practitioners of so-called software piracy.

The public domain has much useful software. The main problem with the public domain is that it is quite vast. Finding what you want, and what you will be able to use is a trying proposition. Also, there are those who delight in "poisoning the well" for others by putting garbage software into the public domain where it can trap the unwary. Garbage software either doesn't do anything, or does something stupid, or even does something (deliberately or inadvertently) destructive, such as wiping out the contents of your hard disk drive.

If you get your freeware from systems and groups that filter the programs rather than just passing on any free program that falls into their hands, you run little risk of becoming a victim of destructive freeware. Such groups will also be a good source of information about what freeware works and what doesn't. Finally, such groups usually don't let commercial software from which the identifying and copyright information have been stripped into their software library in the first place.

Two widely available shareware terminal packages for the Macintosh are:

Pretty Good Terminal

Provides MacBinary and Xmodem transfers, VT100 emulation. (See comparison chart and notes in Chapter Six.) Author requests \$15.

Philip Zimmermann
440 South 45 Street
Boulder, CO 80303

Termworks

Handles MacBinary, regular Xmodem. Includes a dialing directory and macro facility (up to 10 at a time). Author requests \$20.

Horizon Software
c/o James Rhodes
401 Eastwood Place
Lufkin, TX 75901
(409) 637-1455 (Voice)
(409) 637-2286 (BBS)



FILE Menu

Now let's get on with our tour. Throughout the rest of this chapter, words that are in boldface capital letters are MicroPhone menu titles, like this: **FILE**. Words that are in boldface upper-and lower-case letters are MicroPhone menu choices, like this: **Text Capture**. Figure 4.1 shows the MicroPhone menu bar with the **FILE** menu pulled down. The **FILE** menu consists of five sets of subchoices grouped as follows:

- Settings file selections
- Text capture selections
- Printer control selections

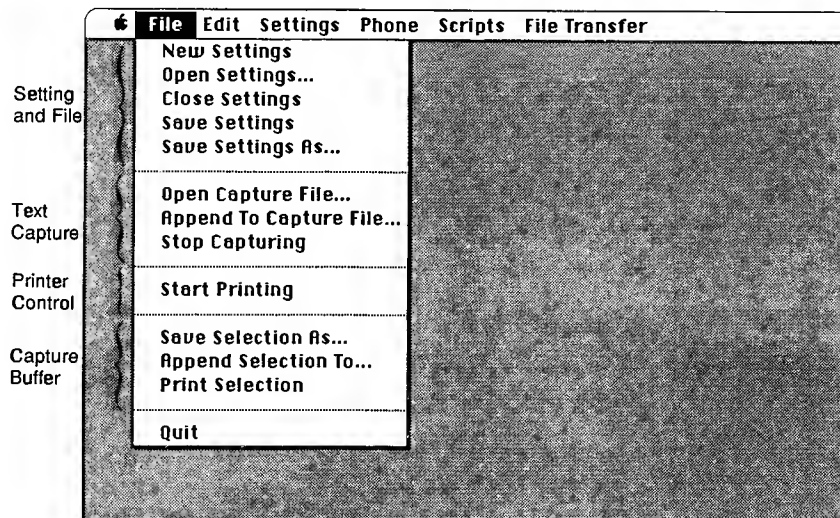


Figure 4.1 FILE Menu

- Capture buffer control selections
- Quit

Settings Files

Settings Files are files that contain information associated with each of the particular systems to which you connect: the phone number, bps, and so on. Returning to our earlier example from Chapter Two, when our contractor communicates with her client, she only needs to open the file she calls UpStart to begin the process. The UpStart file contains the client's computer phone number and the various settings required by his system, including the *log-on procedure*, a sequence of exchanges performed at the beginning of a session to set up the connection. She created this file and customized it the first time she had to establish a connection with the client's computer.

Just as every culture has rituals for establishing contact and disconnecting (saying hello, the handshake, the hug, the ritual kiss on both cheeks, saying goodbye) so do computer-to-computer communications. The log-on procedure creates the context for the exchange of information that follows. The log-off process is an orderly shutdown of the link.

Figure 4.2 shows the "handshaking sequence" for a human-to-human exchange and a Macintosh-to-Macintosh exchange.

Specifications for what are called communication parameters or communication protocols are kept in the Settings boxes. (See Figure

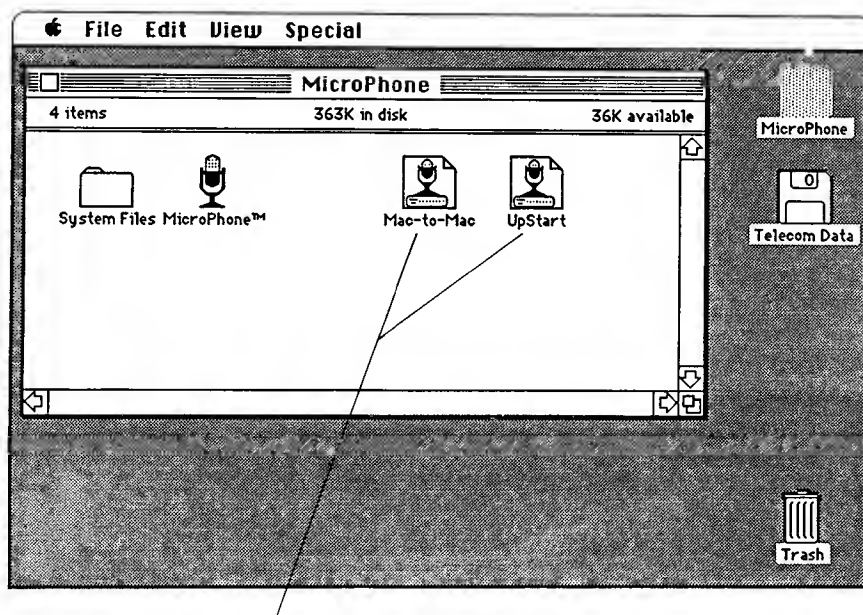
PARTY	HUMAN-TO-HUMAN COMMUNICATIONS	MAC-TO-MAC COMMUNICATIONS
A:	Dials Party B. Waits for voice answer from Party B.	Dials Mac B. Waits for carrier signal (modem tone) from Mac B.
THE PHONE RINGS ("Ring! Ring!")		
B:	Hears ring and picks up the phone. "Hello. Millicent Higgenbotham's residence, Amanda speaking." Waits for caller to respond.	Mac B detects ring and picks up receiver. Sends carrier signal. "eeeeeeeeeeeeee." Waits for first character(s).
THE CONNECTION IS MADE		
A:	Verifies connection: "Hello?"	Reports carrier signal detected. Sends two carriage returns: "00001101 00001101"
B:	Asks who's calling: "Who's calling, please?"	Sends prompt: "Please enter user ID*:"
PARTY A LOGS ON		
A:	Identifies self: "Hi. This is Jane Withers."	Sends ID* to identify account: "123456789"
B:	Asks for verification: "What's your Mother's maiden name?"	Asks for user verification: "Password:"
A:	Responds to verification: "Amanda! It's me, Jane!"	Sends password for verification: "....." [sorry we can't show password]
THE PARTIES GET DOWN TO BUSINESS		
B:	Queries for request: "Oh, yes Jane. How can I help you?"	Prompts for command: "Do you wish to send (S) or receive (R) a file?"
A:	Makes request: "Is Audrey there?"	Enters command: "R"
B:	Responds to request: "Audrey who?"	Prompts for filename: "Filename:"
A:	Responds to response: "REALLY, Jane!"	Enters filename: "Audrey-Notes"

Figure 4.2 Human and Computer Communications

4.3.) You can think of *communication protocols* as the prearranged agreements between two systems that allow them to interpret and use one another's signals.

Text Capture

Files that are moved around (sent and received) we'll call transfer files. The **FILE TRANSFER** menu and the Text Capture section of the **FILE** menu are both used, once you've connected to the other computer, to send and receive files using one of the methods we'll get to shortly. By



These MicroPhone documents contain the proper communication settings to connect to other systems.

Figure 4.3 Settings File Icons

setting up a text capture file and turning it on and off through the menu, you can control which parts of your communication sessions get saved on disk and which do not. Note, for now, that the text capture files are different from the files you may receive through one of the File Transfer receive choices.

SETTINGS Menu

Between them, the **SETTINGS** menu (Figure 4.4) and the **PHONE** menu (Figure 4.13) handle all the behind-the-scenes action necessary for linkup. MicroPhone supplies a number of on-disk Settings files for the more common telecom services, such as CompuServe and Delphi. (See Appendix C.) Whenever a system is contacted for the first time, a new Settings file can be created for the new service. Thereafter, you only need to double-click on the service document from the Finder desktop to contact the new service.

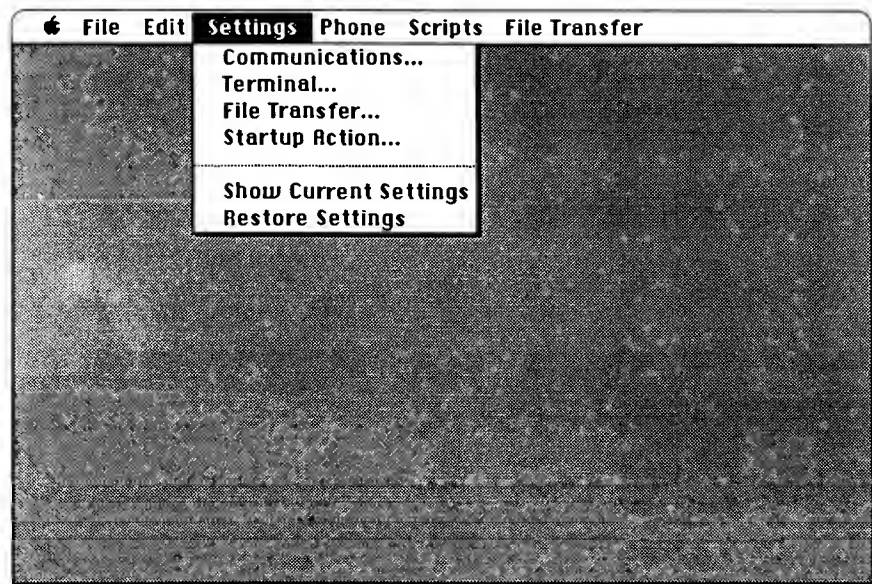


Figure 4.4 SETTINGS Menu

The **SETTINGS** menu has four choices that lead to dialog boxes:

- **Communications . . .**
- **Terminal . . .**
- **File Transfer . . .**
- **Startup Action . . .**

Communications

Figure 4.5 shows the Communication Settings dialog box. The radio buttons select character transmission speed (“Baud Rate”) and character framing or how the letters and numbers will be represented in signal form (Bits per Character, Stop Bits, and Parity Bits). The box also lets you specify whether you’ll be sending data through the modem port or the printer port of your Macintosh.

ASCII Just for a minute, imagine that you and I are two prisoners in adjoining cells with a thick wall between us. There is a small pipe, a tad larger than the diameter of a ping pong ball, running through the wall and opening into each cell (Figure 4.6). We can’t talk to each other by voice through the pipe because the guards will hear us. Fortunately, we

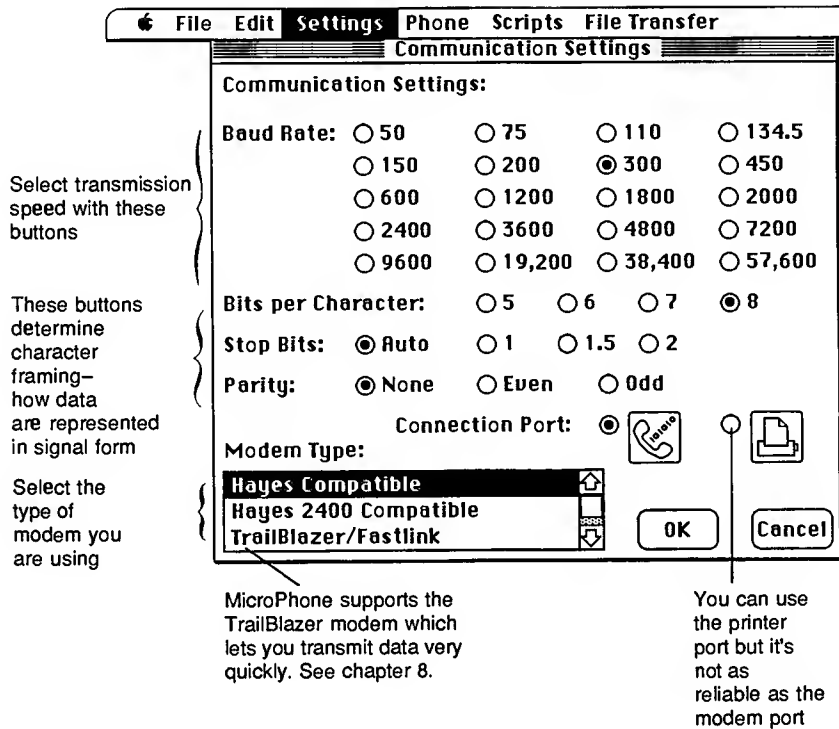


Figure 4.5 Communication Settings

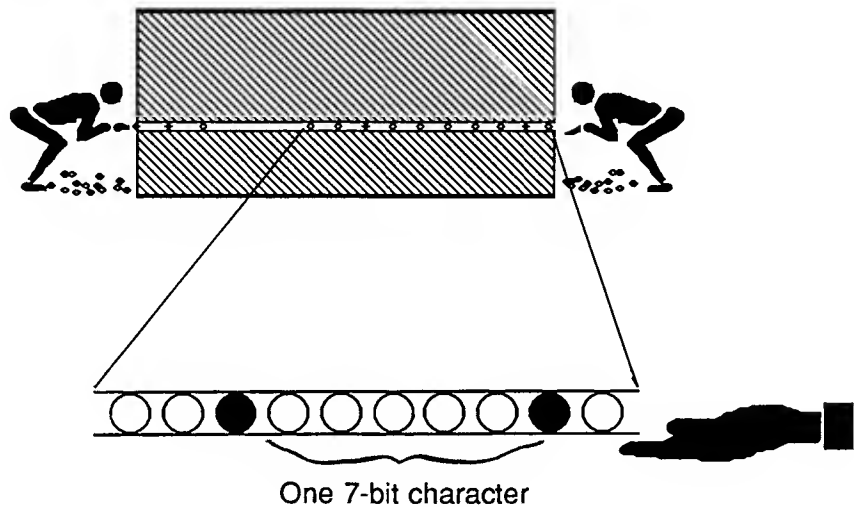


Figure 4.6 Prisoner's ASCII

have a collection of used ping pong balls that will just fit through the pipe. We have a *lot* of ping pong balls, and our task is to work out a way to communicate through the pipe using the balls. In other words, we have to come to some arrangement about what our *communication protocols* are going to be. These will include the physical level (the balls and pipe) and the data level (how many balls will represent one character, and so on).

During our exercise breaks, we develop our scheme. First, we color half of our collection of balls black. We now have a way to represent zeros (white) and ones (black). Each ball represents one unit or *bit* of information.

Next, we agree that our balls will be put together in groups of seven to represent letters of the alphabet, numbers, and punctuation marks. As it turns out, a group of seven balls gives us 128 combinations—enough characters to include the alphabet in upper- and lower-case letters, numbers, various punctuation marks, and a few additional communication control characters. We'll call our code of seven combinations an American Standard Code for Information Interchange, or just ASCII (ă's ' kē).

With our protocols set, all we need to do is send our ping pong balls through the pipeline in the proper sequences and we can communicate. We'll call the number of balls going through the pipe each second the *bit rate* of our communications system. Admittedly, the number of balls we can pump through each second will be small, but we prisoners have a lot of time on our hands.

Everything you'll encounter in telecom between computers is a variation on this basic theme of pipes or *conduits* and ping pong balls. Computers only know “on” and “off”—ones and zeros. With the proper agreements between us, we can use Macintoshes to turn ones and zeros into electrical signals and from electrical signals through wires into meaningful information.

If there is anything approaching a universal agreement or standard in the world of communicating computers, it's ASCII (or sometimes USASCII: the United States of America Standard Code for Information Interchange). It has been dubbed the Esperanto (universal language) of computerdom. Note that we said “approaching.” If it were, in fact, universal, there would be no Bits per Character option in the Communication Settings dialog box. (We've provided a set of ASCII charts and additional information in Appendix B.)

Strictly speaking, ASCII is a seven-bit code. However, in telecom and inside most microcomputers, bits are handled in groups of eight. Hence, most of the time, leaving your terminal program's Bits per Character setting at eight will do the job.

If the system with which you are communicating employs *parity checking*, then the bits per character may have to be set at seven.

The two most-commonly used telecom bit-frame settings are:

No Parity, 8 data bits, 1 stop bit

Even Parity, 7 data bits, 1 stop bit

If you have trouble getting meaningful information on your screen, and if you're not sure what combination the other system is using, try each of these settings first before checking any other part of the connection system to find out what's wrong.

Baud The baud setting you select will depend on:

- The maximum speed the other system is capable of handling
- Whether you are communicating through a direct cable link or through a modem
- If communicating with modems, the speeds at which your modem works (For example, with the Apple modem you can use 110, 300, or 1200 bps.)
- How well the phone lines are performing at any given time
- Whether or not the other system's software can keep up with the maximum speeds of the associated modems

Generally, the speed you select will be the maximum speed that both your Macintosh and the other system are capable of handling. However, under some adverse conditions, you may have to slow down the speed deliberately. A noisy phone line, for example, might be overcome by slowing down the exchange. (See the discussion of baud and bps in Chapter Three.)

Many modems will operate at somewhat lower or somewhat higher speeds than the standard rates of 110, 300, 1200, 2400, 9600, and so on. For example, 450 baud can be used with Hayes Smartmodem 300s with reliable results (on good phone lines). This is a 50% improvement in speed over 300 baud.

However, the Hayes modem just mentioned will not respond to commands you give it (either directly or through your communications software) at 450 baud. In order to take advantage of the higher speed, you must first connect to the other computer at the standard speed.

Then both computers have to be changed to 450 baud. How you tell the other computer system to make the change depends on the other system's software setup. Some computer software won't let you do this, period; other programs, such as MicroPhone and Red Ryder, will. Many IBM PC-based remote bulletin board systems (RBBs) offer the 450 baud option through their utilities menu.

By racing your modem beyond its rated maximum, however, you can expect more errors to creep into your bit stream. Noise may hit the telephone lines and clobber a zero into looking like a one. For the exchange of ordinary text that usually isn't a problem. But for exchanging programs or sensitive spreadsheet or financial modeling data, you'll need to use an *error-checking protocol*. (See Chapter Eight, "Error-Checking Transfer Protocols.")

Another use for a nonstandard speed is to use it as another filter when set up to receive calls. For example, say you want a friend to dial into your Macintosh (after you've set it up to accept calls from other machines) but you want to make sure no one else can log on. You've got a password set up, of course. By using 450 baud and telling your friend that that's what you are going to do, you can thwart those who might try to get into your system using successive password tries.

If the unwanted caller is one who happens to have read *MacAccess* and knows about the "old odd-baud trick" as we described it here, then this security measure will probably fail.

Parity

What is called Parity on MicroPhone's Communication Settings dialog box is referring to the parity bit of each transmitted character.

Parity is equality. In computerdom, and especially in telecom work, to check for parity means to check for equality.

There are many forms of parity checking. Most include some version of a *checksum*. Simply put, a checksum is the number that results when all the bits in a given unit are added. The unit or *block* (or sometimes packet) of bits used for checksumming depends on the particular mathematical method being used and how that method (algorithm) is implemented in a given error-checking scheme. (More on error checking, security, and the Xmodem protocol in Chapter Eight.)

The simplest kind of parity check takes place at the character level. In telecom, when people talk about odd parity, even parity, or no parity, they are talking about a simple form of error checking associated with each character being sent.

Remember that ASCII is a seven-bit character code and in most telecom and computer work, this code is embedded in a field of eight bits. The eighth bit is sometimes called the *high bit* or *high-order bit* because its absence or presence is the difference, in binary terms, between 128 and 256. It's the leftmost bit in the field when you see it written down. For example, here is the letter A in ASCII, in an eight-bit field, with the high bit set to 0:

high-order bit → 01000001

That extra bit can be used to carry additional information about the character, as in the following example.

First, we both agree as to whether we will be using odd or even parity. For the sake of argument, let's say that we're using odd parity. The parity bit that we add in the eighth position of our bit stream, then, will be whatever is needed to keep the sum of all eight bits an odd number. Again, using our ASCII A, here's the character with odd parity:

11000001

Without the parity bit, the sum of the bits is two, which is an even number. To make it odd, we add the parity bit of one, which, when added to two, makes three, an odd number. If we had agreed to make parity even, we would have placed a zero in the high-bit position.

If the receiving computer uses parity checking on incoming characters, it will perform its own addition on the data bits and check to see that the parity bit it comes up with and the parity bit sent with the character match up. If they don't, it's likely that a transmission error occurred.

The main drawback to this method is that if a single bit in a character gets changed, perhaps because of a noisy phone line, the likelihood is that other bits in the same character will also be changed. Therefore, it is possible for the parity to match at both ends even though bits were changed during transmission, but there are more powerful methods for ensuring error-free transmission.



Parity

Most systems you'll be hooking up with will use eight bits per character and ignore parity (the None selection on the MicroPhone Communication Settings dialog box). If the remote system requires seven bits per character, it is likely to expect parity bits, too. Seven bits per character, even parity is your best guess. If that's wrong, you only have to try one other setting: seven bits, odd parity.

Most of the time, you will have to set your software to conform to whatever the other system expects. In the majority of cases, parity checking won't be used. The other system may ignore the high order bit or set that bit to zero when exchanging ASCII text. When you are not sure what the other system expects, begin with None for parity and send an eight-bit character.



Mark and Space Parity

You may also run across the terms *mark parity* and *space parity*. Parity as we've just explained has little to do with parity as used here.

Mark and space are other names for the modem tones representing zero and one. The *mark tone*, then, is the tone generated by the modem which represents a binary one. A *space tone* is the modem tone representing a binary zero. *Mark parity*, therefore, means to always set the eighth bit to be a one (high). *Space parity* sets the eighth bit to zero (low). Why the term parity is used at all in this situation is beyond the scope of *MacAccess*. You generally won't have to think about it. Just remember that the *idle state* of a modem means that a continuous mark tone is being sent (a one) and that a space tone is equivalent to a zero. Zeros are also used as start and stop bits. So, when we say that the modem or line is in idle, we mean that there is a signal representing a one on the line. A start bit is sent, which means that the line switches to a space tone, and the receiving computer begins counting bits and building up a character.

Asynchronous Communication Throughout most of our discussion in *MacAccess*, we are concerned with *asynchronous* (sometimes shortened to *asynch*) communication as opposed to *synchronous* communication. Synchronous communication requires synchronous

modems at both ends. In asynch communication, the interval between signals, in this case between characters sent, is undefined and varies over time. For example, when you are typing letters on your keyboard, the length of time between characters varies according to your typing speed, the particular letter combination involved, and how good or bad your manual dexterity is on a given day.

When working on line, there is usually a signal dialog of some kind going on between your Macintosh and the computer at the other end. The number of characters sent and the interval between those characters will vary over the length and purposes of the session.

At the minimum, each computer must know where the beginning of a character is in order to properly frame the incoming bits; therefore, a *start bit* is always sent. Then the data bits are sent (seven for ASCII). Then the parity bit is sent. (If none is required, this bit is sent anyway, usually set to zero, and then ignored.) Finally, one or more *stop bits* are sent, signaling the end of the character and putting the modems back into a rest or idle state to await the next start bit.

Sometimes a line is said to be in the idle state. When two systems are connected, the line is idling when the tone representing the binary digit one (the mark tone) is being sent continuously. The system of lines and computers is said to be in the *wait state*. During the wait state, both computers are waiting for a start bit.

The start bit is represented by a second tone: zeros (space tones). The start bit tells the other system to begin “watching the clock.” The time each zero or one tone lasts is used to calculate how many bits were sent. For example, at a transmission rate of 300 bps, a single bit lasts for 3.33 milliseconds. If the tone representing a one lasts for 6.66 milliseconds, then the system knows that two bits in a row have been sent. Stop bits are represented by mark tones—just like the idle state. For a detailed explanation of the tones and their specifications, refer to one of the technical references in the Bibliography, such as *RS-232 Made Easy*.

Note: Two stop bits are required for 110 bps communication. When you select this speed on most terminal programs, the additional stop bit is added automatically.

One and one-half (a stop bit that lasts one and one-half times as long as a single stop bit) stop bits are used with 134 bps, usually with a mainframe that thinks it's communicating with an IBM 2741 Terminal, which is about 15 characters per second—slower than a human typist.

Most asynchronous communication takes place with the bits arranged in groups of 10. Figure 4.7 shows the letter A again with one start bit, seven data bits, a parity bit, and one stop bit.

When Auto is selected, MicroPhone will set the number of stop bits to 2 for transmission speeds from 50–110 baud. At speeds above 134.5, 1.5 stop bits will be used. All other speeds will use 1 stop bit for the default value. You can override these settings by choosing 1.5, 1, or 2 stop bits from the dialog box.

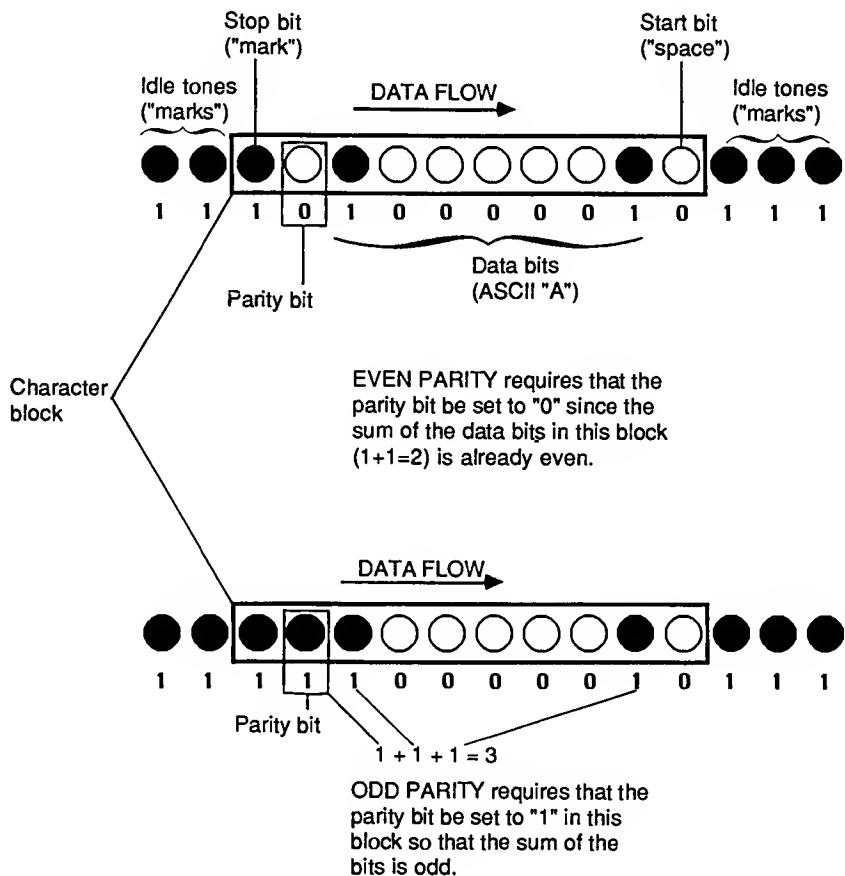


Figure 4.7 Character Block of 10 Bits

Connection Port You can connect a cable to either the printer or the modem port. A connection port selection, then, is an essential part of any communications program. It tells the software where it should send and look for signals. For most communications work, the port of preference is the modem port, especially if you want to achieve quick transfer when direct-connecting by using a higher baud setting (say 57,600) when connecting directly to another Macintosh.

Under some conditions you might need to use your printer port instead. For example, you might be already using your modem port with a hard disk drive. See "Interrupt" (Chapter Three) for some suggestions for using the printer port rather than the modem port for telecom.

Modem Type Hayes Compatible should be selected if you are using a Hayes modem or a modem (up to 1200 bps) that supports the Hayes command set, such as the Apple Personal Modem. The Hayes 2400 Compatible selection is the choice to use with that modem, or with modems that also communicate at 2400 bps using the Hayes protocol for high-speed data exchange. Finally, if you are one of the avante garde who are using the Telebit Trailblazer, select Trailblazer/Fastlink.

Terminal Settings

TTY A common denominator in the telecom world is TTY. It is a little like pulse dialing—a throwback to an earlier, more mechanical, less electronic world. The teletypewriter or TTY was once state of the art. There was a time when no major company worth its salt didn't have a TTY clattering away in the back room. The military alone accounted for sagans* of them.

A teletypewriter is just what it sounds like: a mechanical typewriter or printer at the end of a wire. Signals came in on the wire and got printed out, line-by-line, on long rolls of paper by the TTY.

For the historian who studies the development of technologies, the past is buried in our current ways of doing things, no matter how sophisticated we may think today's techniques are. Nowhere is this fact more evident than in the field of telecom. The fossils are not far below the everyday surface. Many ancient artifacts are still widely used, which is why today's generation has to put up with yesterday's "standards."

*Billions and billions

For historical and practical reasons, then, about 80% of the telecommunicating going on today is influenced by or still uses the developmentally ancient ASCII-TTY standard, only faster. Just in case you ever find yourself having to communicate with a *real* TTY machine, remember to slow your terminal program down to 50 or 110 bps, and add two stop bits.

The word Teletype is sometimes used to refer generically to teletypewriters. Teletype, however, is a trademark name for a specific brand of teletypewriter, the Teletype ASR-33, popular in the sixties and seventies. The ASCII code was developed, in part, to accommodate the need for a standard method of communicating between different makes of teletypewriting machines.

TTYs are *line oriented*; characters are sent one line at a time. At the end of each line, a carriage return signal (CR = ASCII 13) and a linefeed signal (LF = ASCII 10) set the TTY to the next line, ready to print another line of text.

When you select TTY on the Terminal Settings dialog box (Figure 4.8), you are making your Macintosh behave like or *emulate* an earlier technology, the teletypewriter.

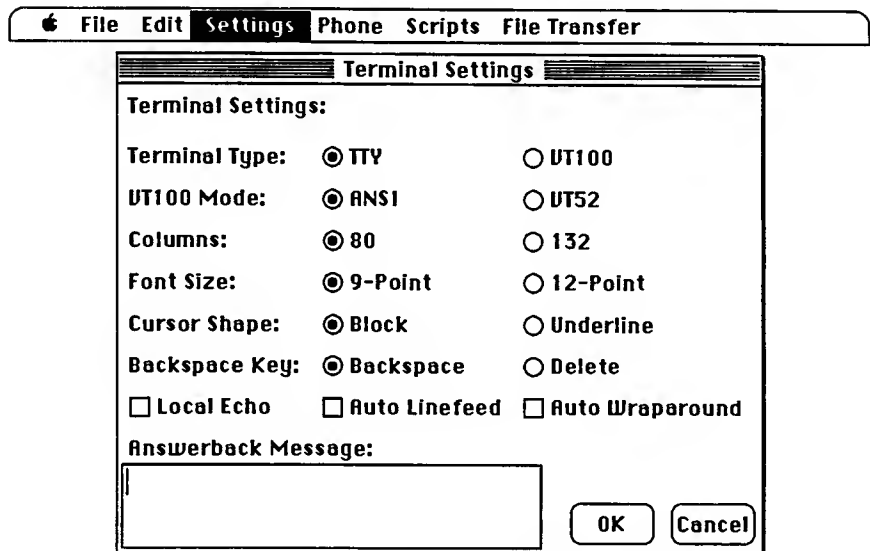


Figure 4.8 Terminal Settings

VT100 MicroPhone on the Macintosh can also emulate VT100 and VT52 terminals. These settings are used with DEC minicomputers and mainframes, as well as others, and are fully *screen-oriented*, which is several steps beyond line-oriented communications, more suitable for text on a television screen.

Since TTY emulation is only concerned with lines, any command that is screen-oriented, such as “clear the screen” or “cursor up,” won’t work. Practically speaking, this means that most TTY-oriented terminal programs (emulators) can’t do full-screen editing or use special graphics characters to draw pictures on the screen. This ability is frequently called *full-screen cursor addressing* because the cursor can be positioned and repositioned to “draw” graphics and alphanumeric characters on the screen.

Such screen-oriented emulators allow things like changed typeface, blinking or reversed characters, and the whole gamut of tricks possible on the television screen. That’s what VT100 is all about and where the Macintosh shines.

Local Echo Figure 4.9 shows a Macintosh connected to another computer, computer B. Recall from our discussion about the RS-232 standard in Chapter Three that *full-duplex* (sometimes just *duplex*) means that signals are going in both directions at once over the communications link. If only one side of the link has use of the same channel at a given time, it’s called *half-duplex* communication.

In most of your Macintosh telecom work, you’ll be using full-duplex communication. In fact, very few of the systems you can connect with are true half-duplex communicators. The difference between *full-duplex* and *half-duplex*, as far as most communications software packages use the term, is whether or not the remote system echoes characters; that is, whether the remote machine will send back any character you send so that the character is displayed on your screen. This is represented schematically in Figure 4.9a. This is sometimes referred to as *echoplex* communication. Echoplex is an elementary form of error-proofing, since you won’t see the character you typed on the screen unless and until the other system actually gets it and echoes it back.

If, however, the system you’re linking up with asks for half-duplex communications, then it probably simply means that it will not echo

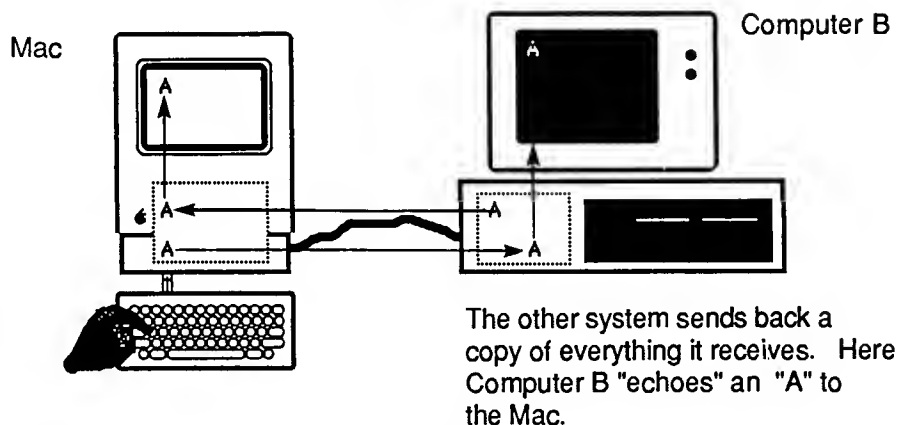


Figure 4.9a Full Duplex with Remote Echo

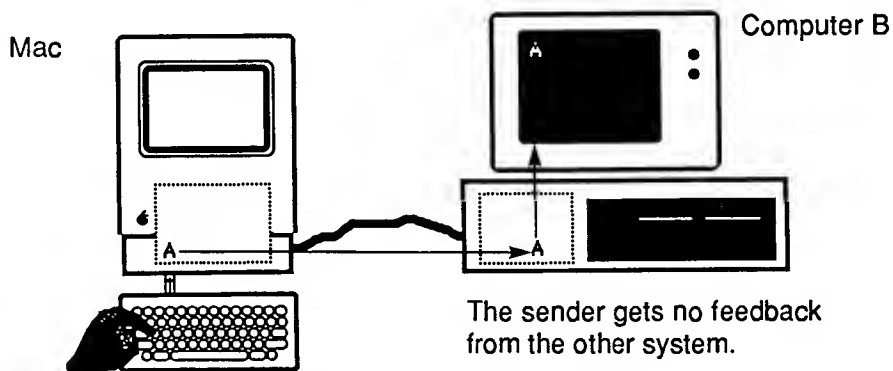


Figure 4.9b Half Duplex or Full Duplex without Remote Echo

your keystrokes, and you will see nothing on the screen unless you check the Local Echo box. This is represented schematically in Figure 4.9b.

With Local Echo set, your Macintosh will send the characters you type to the screen and out through the modem at the same time. This condition is represented in Figure 4.9c.

If you see double characters when you type, but only single characters when the other system sends you data (as shown in Figure 4.9d), then you probably have Local Echo checked and are seeing the results of two echoes: one locally, from your Mac, and one from the remote system. Turning Local Echo off will cure your Macintosh of its stuttering.

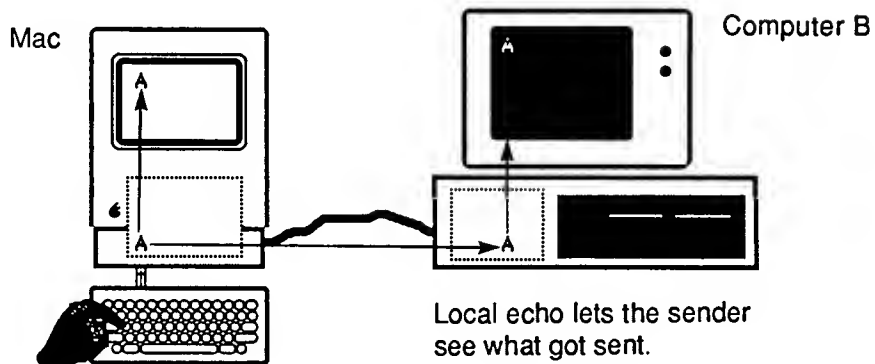


Figure 4.9c Local Echo Only

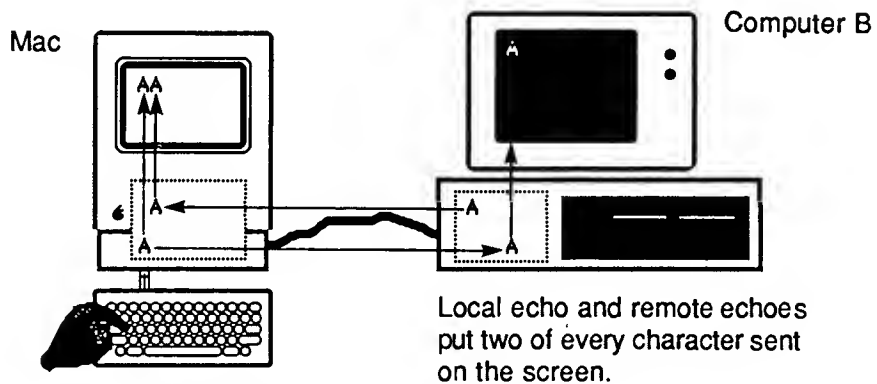


Figure 4.9d Local and Remote Echoes

True duplex or full-duplex communication (two computers sending and receiving a stream of bits at the same time) is rare in the world of micro telecom. A channel may be capable of full-duplex communication, as the telephone system is, but the machines on either end of the connection simply may not be able to handle a full-duplex exchange, not because of the machines or the phone lines but because of the software being used.

Most error-checking file transfer protocols (such as XModem, Chapter Eight) perform their work in half-duplex fashion. They are said to be *half-duplex protocols*. There are exchange protocols that allow two systems to exchange files simultaneously. These make sense, for example, in transnational situations where high volumes of data need to be exchanged both ways on the link. See Chapter Six, "Micro to Mainframe Software Solutions," for more about one of these protocols called BLAST.

Auto Linefeed and Auto Wraparound In line-oriented, TTY-style communications, it's the job of the sender to insert linefeeds and carriage returns at the end of lines. It is assumed, in most cases, that the lines are 80 columns wide, however 132 is also common. Normally, the CR (ASCII 13) is sent, followed by a linefeed character (ASCII 10). This sequence is sometimes written in print as CR/LF.

If, for some reason, the other system fails to send a linefeed, then you'll have to instruct MicroPhone to put them in for you. The sign that you need the Auto Linefeed box (Figure 4.8) checked is that lines overprint, one on top of the other, and do not scroll down the screen.

The Auto Wraparound feature is similar. If the other system sends you an unbroken stream of characters without any CR/LF sequences at all, or if it sends you a line longer than you can display without having it "fall off" the right side of the screen, then you'll need to instruct MicroPhone to automatically break the long lines as they are displayed.



Carriage Returns and Linefeeders

This business of carriage returns and linefeeds can become very annoying. In some files, you'll want a carriage return only at the end of paragraphs. In some files, you won't want any "hard" CR/LFs, but, you will want your own word processor to format line breaks according to type font and size. There are various utility and file translating programs that will strip or insert CRs, LF's, or CR/LF sequences. See Chapter Eight.

Answerback Message Some mainframe systems are capable of automatically dialing your Macintosh and sending data to it. In such cases, an *answerback* message may be required in order to establish the link. The Answerback Message box (Figure 4.8) is where you enter the

message the other system will expect to get upon first connecting. It may be something like "logged on gensys 2250 hours" (which might mean that you have logged onto Gengle's system at 10:50 p.m.).

File Transfer Settings

We are not quite beginners at this game, but we appreciate how a neophyte might feel upon encountering the dialog box in Figure 4.10 for the first time. If you have never used a modem before, much less transferred a file from one place to another, you might move around this dialog box forever with your little mouse and not accomplish much. Even if you are comfortable with telecom and already use it a lot, Figure 4.10 may be quite confusing to you.

For the majority of communications tasks, you'll be able to leave this dialog box set at its *default* (factory-preset) settings and ignore it.

Someday, however, you may run into a unique situation not covered by the default settings. Then what do you do? That's why this dialog box is there, after all. Fortunately, it's not as formidable as it looks once you get past the jargon.

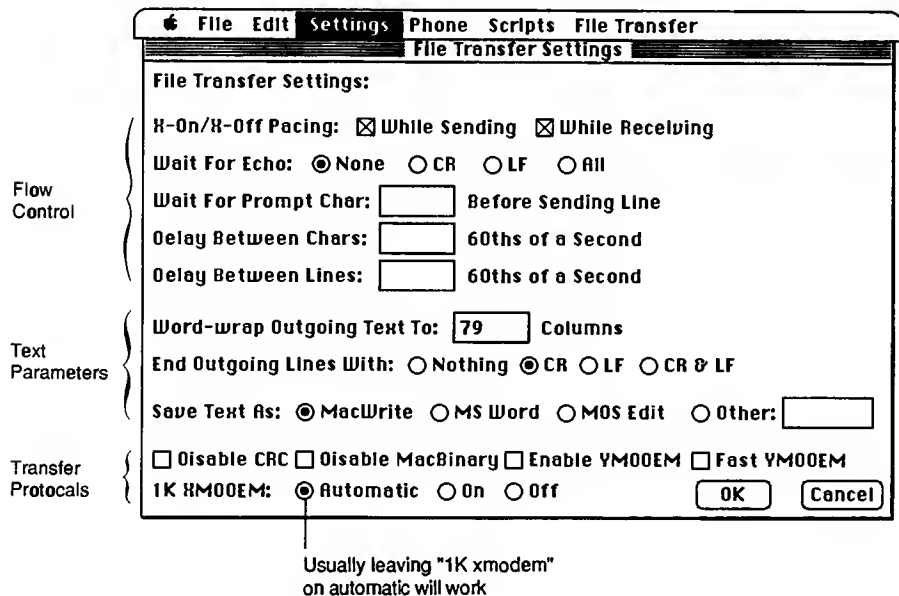


Figure 4.10 File Transfer Settings

X-On/X-Off Pacing Let's say that instead of electricity through a wire or ping pong balls through a pipe, our bit stream is water flowing into your bathtub. Let's say, further, that you are an eccentric inventor prone to getting sidetracked by your active and imaginative mind.

You like to take long hot baths and think about your current problem(s). However, when filling the tub you sometimes forget that the water is running, and on at least one occasion you flooded out the downstairs tenants. (There is an overflow drain, but if the water is running fast enough, the drain won't empty the tub quickly enough to prevent eventual overflow.) On the other hand, if you turn the water off, you may come back 45 minutes later to find the water tepid and not at all inviting.

What to do? You want the water hot when you get there, but you also don't want to make the tub overflow. Figure 4.11 shows the solution. Water sensor A is placed just above the overflow drain on the bathtub. It is connected to valve B, which mixes hot and cold water in just the right proportions. It is either open or closed.

Now you can turn on the system, set the water running, and leave knowing that the hot tub of water will be there waiting whenever you want to go back to it. The water flows through the pipes into the tub. When the water reaches level 2, the water level sensor turns off the water and the tub continues to drain through the overflow drain.

Conversely, when the water drops to level 1, the water sensor turns the water back on, replenishing the water level and temperature.

This is what *flow control* is all about: making sure that the steady transmission of bits doesn't overflow and overwhelm the receiver, resulting in annoying errors.

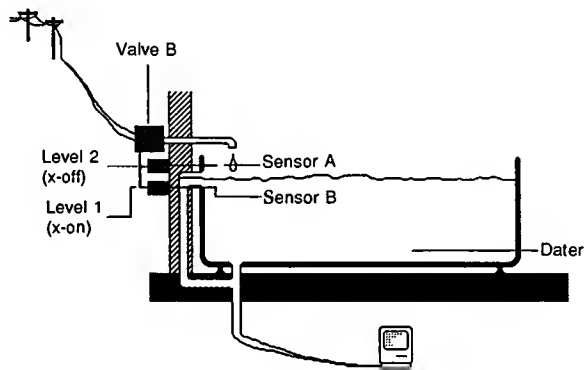


Figure 4.11 Flow Control

X-On/X-Off pacing is one of the most common methods of flow control. The principle behind it is similar to our eccentric bathtub example. The bathtub is really a *memory buffer* inside your Mac's RAM. In the parlance of the software engineers, a *buffer* is any temporary memory storage area. A buffer is necessary in order to compensate for the differences in speed between the primarily electronic world inside the computer and the physical world of disk drives and printers. In our case, we're concerned with getting bits from the phone lines onto a disk for more-or-less permanent storage.

As bits come in through the modem, they are put in the buffer. When the number of characters or bits stored in the buffer reaches a certain point (usually somewhat below the point of actual overflow), the receiving computer sends an X-Off character, which is actually ASCII 19 (Cntrl-S), to the sending computer, which then turns off the flow of characters and waits for the go-ahead, X-On (ASCII 17, Cntrl-Q).

The receiver can send X-Off for other reasons. For example, if you are reading information while on line, by manually sending the X-Off character you can stop the information from scrolling off your screen long enough to read it. Sending X-On manually will then start it up again. Your Macintosh may also send the X-Off signal in order to write the contents of the incoming information buffer onto the disk.



X-On/X-Off

Many systems have implemented X-On/X-Off. To simplify flow control for on-line users, some systems now use S or Cntrl-S or s (three different ASCII characters) to temporarily stop transmission. If you're making a new connection to a system you've never used before, one or more of these are likely to work. Additionally, some systems also allow you to restart transmission by pressing the S key again (sometimes called *toggling*). Other systems let you restart by pressing any key, the space bar, a return, or sometimes the *real* X-On (Cntrl-Q). A little experimentation will tell you what works if you don't have access to documentation for the system with which you're connected.

Sometimes a few characters may be sent after you've already sent the stop signal. It takes time for your signal to reach the other computer and for the other computer to react by shutting off the flow. In that intervening short period, several characters may arrive.

Some systems only support X-On/X-Off pacing in one direction, while sending. In any case, by clicking on the appropriate squares in the File Transfer Settings dialog box, X-On/X-Off pacing can be controlled.

Wait For Echo Be careful not to confuse this selection with the Local Echo feature, covered earlier. In the present case, the awaited echo is assumed to be coming at all times from the other computer. Once the specified character is received and echoed, then the next action will be taken. The action, in most cases, will be to send the next character or line of text.

To understand this better, assume that two computers are communicating in full-duplex. Your Macintosh is sending characters to the other machine faster than it can catch and store them, and X-On/X-Off flow control may not be available. By setting Wait For Echo to pause and look for a carriage return before sending the next line of text, you can slow down the exchange to compensate for how slow the other system is.

The most often used echo to wait for is CR (carriage return). Again, it is assumed that your communications software is sending information one line at a time to the other system, with either or both a CR and LF (line feed) at the end of lines. Here, by telling MicroPhone to wait for the other machine to echo back the CR, it is unlikely that you will overwhelm the other machine's internal flow control mechanism.

By checking the All choice, you'll really slow down the system, making the receiving computer echo *each character* in turn before the sending computer delivers the next one. Needless to say, this should be used only in a last ditch effort to get the other system to digest your data. If connected in a half-duplex fashion, these options won't work. (In MicroPhone, they will be dimmed if you are connected to a half-duplex system.)

Wait For Prompt Some remote systems, Conference Tree message systems, for example, prompt the remote user for information. That system sends a line number and a short dash (or hyphen) to signify that the other computer can send a new line.

Line 10

-

In this case, you tell MicroPhone to wait for the prompt character (-) before sending a new line.



Setting Your Own Prompts

Some systems let you define what prompt you want. If you get to choose one, try to pick one that is not found in the data you will be transferring. Otherwise, the echo from the embedded prompt character will trigger the sending of a new line before the other machine is ready for it, resulting in lost information.

Delayed Characters and Lines What if the other computer can't use X-On/X-Off pacing and is a half-duplex connection, and you're overwhelming it with all your fresh data? If that's your problem, never fear.

Just make MicroPhone delay transmission by putting extra time between characters or lines. Start out with relatively low values and keep experimenting until you get a successful transfer.



Null Character

The ASCII character set has a special character called a *null* (ASCII 0) that can be used to accomplish time delays between lines or characters of text being sent or received. The null is simply a character that takes time to transmit but doesn't really do anything. Rather, its function is to be to the ASCII character set what a zero is to the number system. Not to be confused with blanks or spaces (ASCII 32), which will put spaces between characters, nulls take up time but no space. Many terminal and host programs let you specify that one or more nulls be sent either between characters or between lines. The most common reason to send them between lines is to give a printer time to return the print head to the beginning of the next line and advance the paper.

Wrapping Words and Ending Lines Let's say you have a text-oriented report that has 132 characters on each line. The computer to which you'll be sending the report only displays 80 characters per line. By entering 79 (79 instead of 80 just to be safe) in the Word-wrap Outgoing Text To window (see Figure 4.10), you can tell MicroPhone to send only as many whole words per line as will fit in the 80 columns.

This selection gives you some control over how your text is being displayed on the other system's screen. If text you send is being captured and stored in a file at the other end where it will be printed out or worked over by a word processor, you may not want to have your system control the word wrap. Further, if the text you are sending consists of a lot of tabular and specially formatted material, say from a Word document, the other system will not be able to use the original formatting of the file.

You have control over how outgoing lines will be ended, too, depending on what the other end does or needs. Not every terminal package lets you control such things. Some supply their own LFs or CRs to incoming lines. Others don't do either, so you'll have to make sure they're sent to you.

In any given instance, you may have to fiddle with the settings available on your terminal package and the second system's controls until you get a combination that gives you satisfactory results; that is, you get results that look readable on screen and/or on paper.

Save Text As Here, the Text referred to is the text coming into your Macintosh. With MicroPhone you have your choice of saving text as MacWrite or Word files, or as MDS Edit (Macintosh Development System; see Appendix C) files. MicroPhone lets you specify other formats by providing the Other box.



Creator Labels

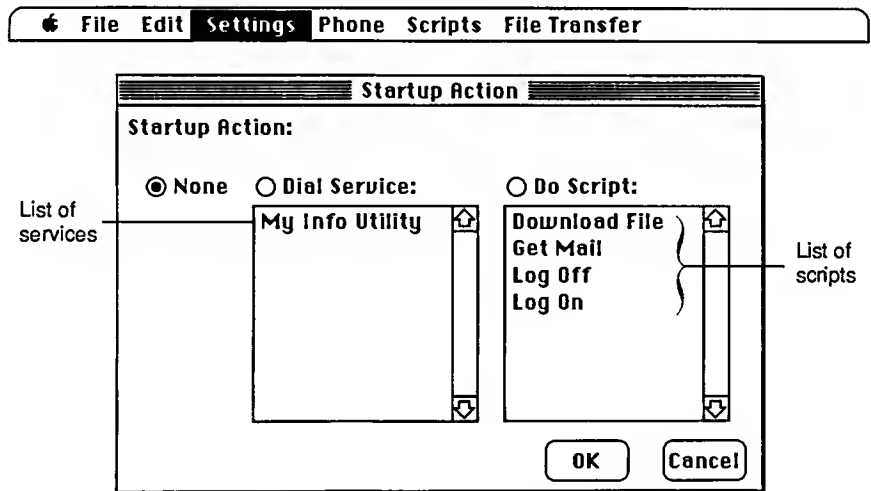
Each Macintosh file has a Creator label in its resource fork which is used by the Finder. This attribute can be found by examining the attributes of an application file (typical of the one you want to transfer) with programs such as Mactools from Central Point Software (Appendix C) or DAFire (Appendix E).

Transfer Protocol Adjustment With MicroPhone, there are three main protocol choices: text, Xmodem, and MacTerminal. These are chosen from the **FILE TRANSFER** menu (covered at the end of this chapter). The protocol you choose will depend, in part, on what you're trying to transfer, and in part on the capabilities of the remote system with which you are connecting. The bottom two lines of the File Transfer settings dialog box let you "fine tune" the Xmodem protocol, if that's what you selected from the **FILE TRANSFER** menu.

The Xmodem adjustments include the option of sending 1K blocks instead of the usual 128 byte blocks. You can usually leave this setting on Automatic and let MicroPhone take care of the matter. If you select "On," MicroPhone will *always* send 1K blocks, regardless of what the other system signals. If you select "Off," MicroPhone will only send 128 byte blocks, even if the second system signals that it is able to handle 1K blocks.

Startup Action

The Startup Action feature is nice to have in a terminal package, especially if you do a lot of telecommunicating. The Startup Action dialog box (Figure 4.12) lets you specify what MicroPhone is supposed



The Startup Action dialog box lets you specify a service to call or a script to run when you open a settings file.

Figure 4.12 Startup Action

to do when you double-click on a MicroPhone document. You can go from the desktop to log on with two clicks of the mouse!

Dial Service lets you automatically dial-up a given system. Once connected, you are given control of the terminal. However, if you wish, you can have the Dial Service selection, in turn, run a script after connecting. The script (MicroPhone's name for a program) can then perform automatic log on, mail retrieval, file transfers, and so on. Do Script will cause a script to be executed when MicroPhone starts up. The script can contain instructions for when to dial the phone, where to dial, how to log on and conduct business, and so on.

Startup Action and its use with scripts is the outer edge of possibilities we'll explore more fully in Chapter Five.



Turnkey Communications

If you have clients who have Macintoshes, you can ease them into telecom, help save them time, and put competitive feathers in your cap as well by creating special *turnkey* systems for them. A turnkey system is one that performs its task "at the turn of a key," the way automobiles are started.

By customizing their copies of MicroPhone for them, using MicroPhone's built-in programming language, you can create a disk that automatically dials your Macintosh and conducts business for your client with little or no intervention. You could also customize an electronic mail package for them. Further, you could manage the whole communication process yourself, remotely, by setting up your client's machine to act as a sophisticated answering machine. You then dial in and take charge of the transfer process. All your client needs to do is know how to use the application for which the file is intended. The possibilities are endless, and all that's required is a willingness on your part to learn a little more about telecom than your clients in order to help them maximize their investment in your services, whatever they may be. Such telecom knowledge can enhance any service you're now performing for hire.



PHONE Menu

Many commercially available terminal software packages provide telephone management tools. In some cases, this takes the form of a directory for entering the names of systems, the phone numbers

where they are connected, and the communications parameters associated with the number. Other amenities can make telephone management (see Chapter Seven) easier. For example, some packages will automatically keep track of how long you are connected to a given phone number. This makes it easier to check your monthly phone and related communications services charges. MicroPhone's **PHONE** menu is shown in Figure 4.13.

Set Up New Service

Strictly speaking, not all the systems you'll be connecting with will be services. For MicroPhone's purposes, a *service* is any other system with which you want to connect. To set up a service (see Figure 4.14) is to tell MicroPhone the name (which will later show up on a pull-down menu), the phone number, and how to dial the phone (whether with tone or pulse dialing, or a mixed number such as a local pulse number combined with a tone number for a long-distance carrier).

You can associate a script with the number, and instruct MicroPhone to play back the script after connecting with the other system.

After you've set up a service and given it a name, its name will be displayed on the **PHONE** menu under Dial Service (see Figure 4.13). The menu also gives you the option of changing or deleting services.

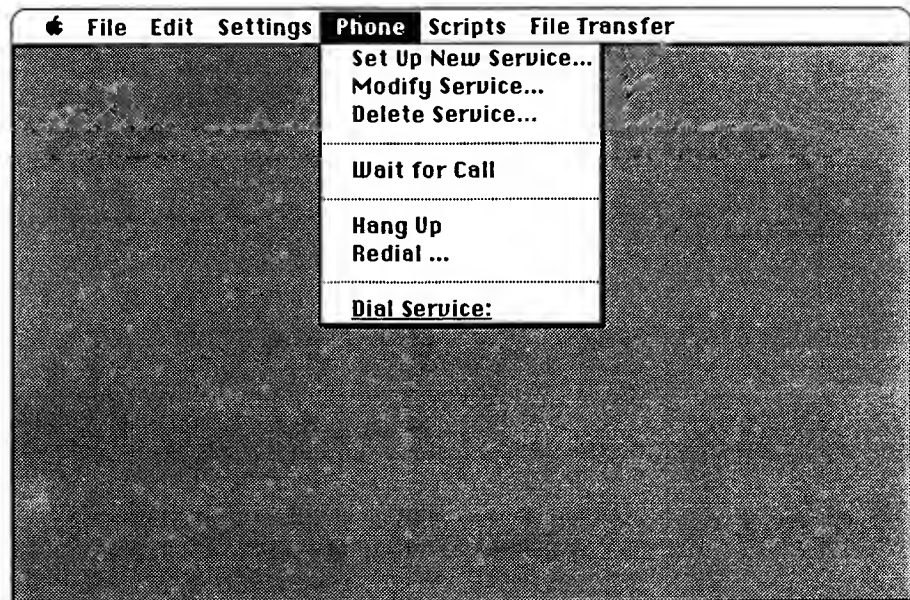


Figure 4.13 PHONE Menu

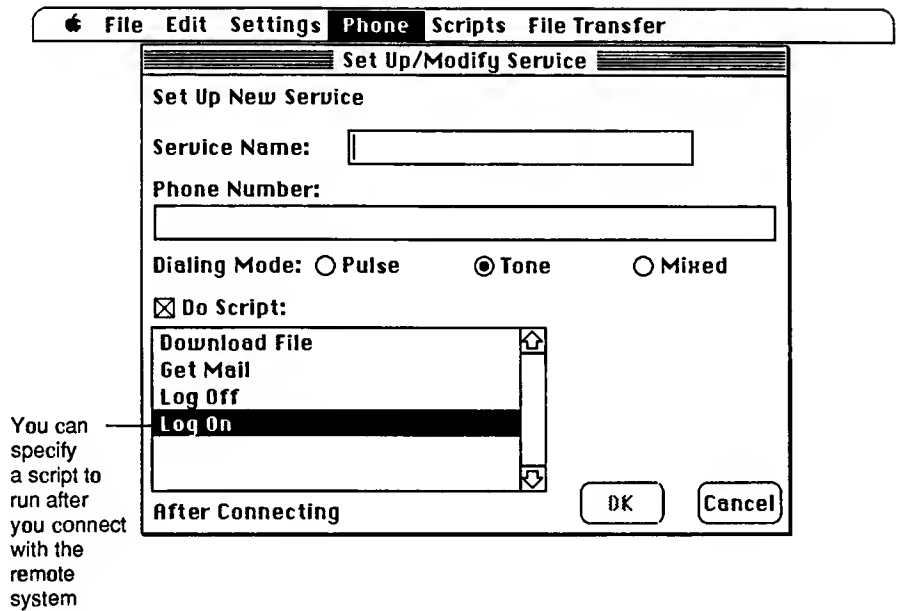


Figure 4.14 Set Up New Service

Hang Up and Redial do just that. Redial will redial the last number tried until you get connected or you tell it to stop (Abort).

Wait for Call makes your Macintosh into a *micro-host*. When a call comes in, MicroPhone will answer and connect with the caller. See Chapter Five for a discussion of hosts and micro-hosts.



Attack Dialing

Many systems, depending on how many users they have and how many phone lines support those users, will get crowded from time to time. If you want to dial into such a system, you usually have to keep trying until you are successful. With MicroPhone, you can create a script that automatically redials the busy system until you get through.

Using Redial this way is called “attack dialing.” This is useful in logging onto busy, single-line public BBSs (bulletin board systems). There are also utilities available that will “attack dial” in the background while you are doing other things with your Macintosh, such as writing your next letter to the editor. As soon as the dialer gets through, it rings bells so you can enter your terminal program and conduct your business.

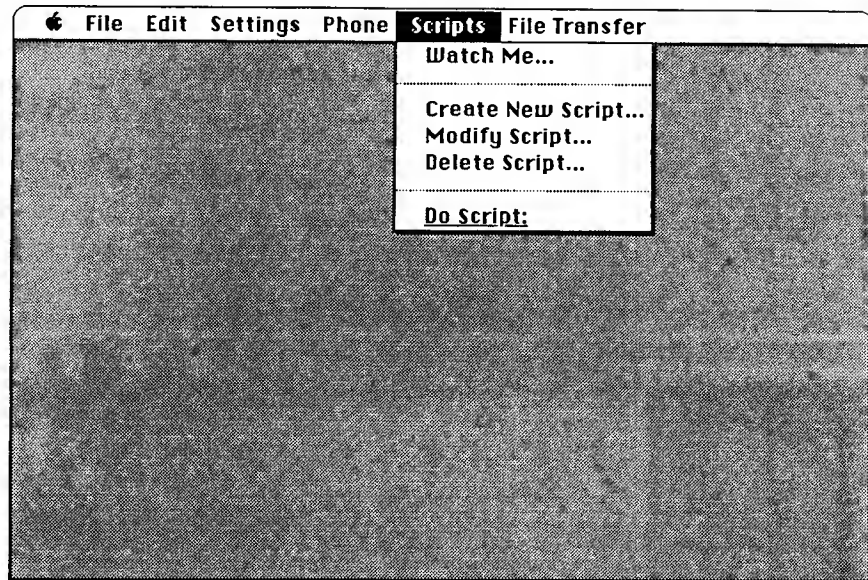


Figure 4.15 SCRIPTS Menu

SCRIPTS Menu

MicroPhone's Watch Me feature (Figure 4.15) uses an idea that is being incorporated into more and more software of all kinds, including other terminal packages. They all function in a similar way.

With MicroPhone, by clicking Watch Me you begin recording your telecom session. You then do whatever it is you wish to do: fetch your electronic mail log onto a new system; send a file to another micro; and so on. MicroPhone captures your keystrokes and the responses of the other system and places them in a script. You then save the script under its own name for later replay. Finally, you can use the built-in script generator to edit and refine your Watch Me scripts, link them together, add control structures, and so on. By refining your recorded scripts this way, you can create sophisticated automated telecom applications.

This feature is so powerful, we've devoted part of Chapter Five to a further discussion of Watch Me.

FILE TRANSFER Menu

We've now come around full circle on our subject and are ready to spend more time with files, the heart of any information transfer. Figure 4.16 shows MicroPhone's **FILE TRANSFER** menu with three Send alternatives and two Receive choices.

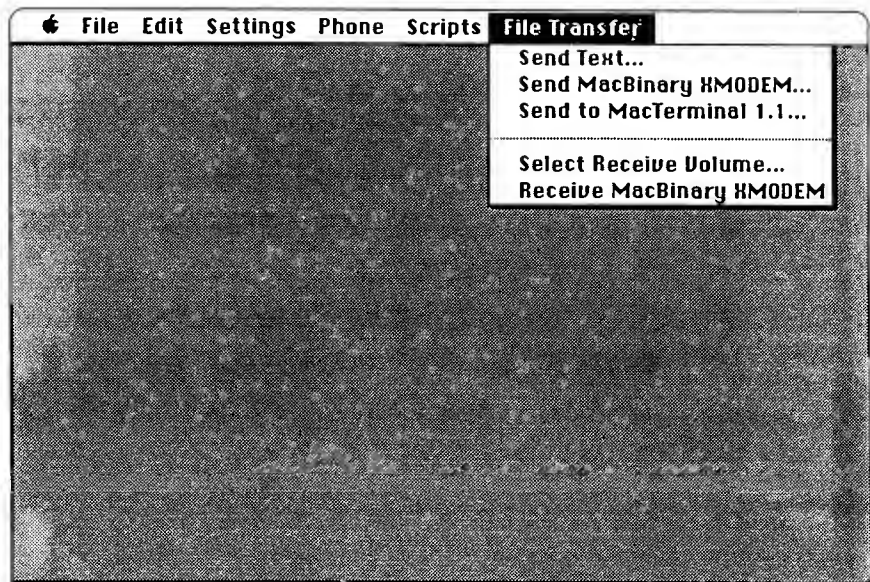


Figure 4.16 FILE TRANSFER Menu

Remember, a Macintosh file is a bundle of bits that has to travel together to be useful. The bundle of bits can represent anything: text, rows and columns of number values for a spreadsheet, records and fields of a data base, program listings written in a particular program language, or programs stored in machine language or binary form. Finally, the file might also contain graphics of some kind, such as PageMaker, MacPaint, or MacDraw documents.

Two factors determine how a given file should be sent, and therefore which Send selection to choose:

1. The contents of a file
2. How the file is going to be used at its destination

Let's take each of these factors in turn using a simple example we'll call the *vanilla text file* transfer (one of the most common kinds of file exchange), and a second, rather more complicated, case which we'll call the Macintosh-to-Macintosh *transparent file* transfer.

Vanilla Text File Transfer

Recall the two forks (Figure 4.17) of the typical Macintosh file: the resource fork contains Macintosh-specific information. The data fork contains everything else.

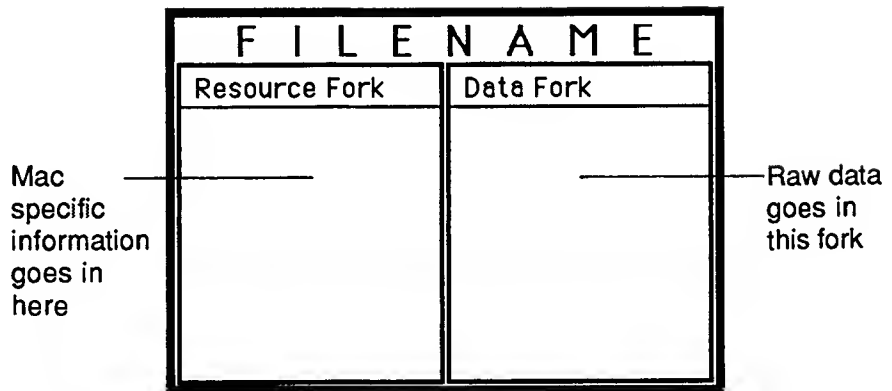


Figure 4.17 Macintosh File Forks

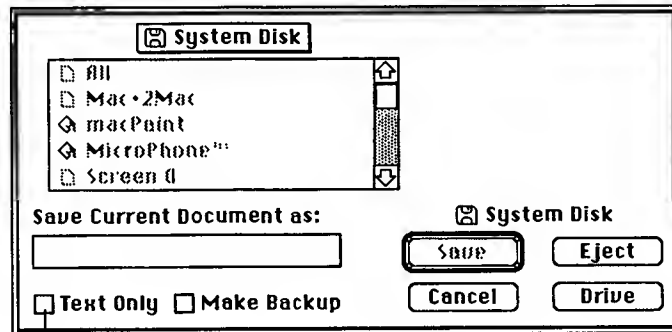


Figure 4.18 Word Save Dialog Box

Let's say you've created a Word document: a draft of a progress report or sales summary. When you are ready to save your document, Word gives you two alternatives as shown in Figure 4.18: to save the document as a fully formatted Word document, or to save it as a Text Only file.

A fully formatted document contains all the font and style information (boldface, underline, italic, and so on), and paragraph-formatting information required by Word to make the document come out right on the screen and on your printer. In addition, the resource fork of the Word document file contains its icon and finder information so that it can be displayed properly on the desktop.

A text-only document won't have any text style information in it (boldface, italic, and so on). In addition, the resource fork will have its file type set to text, so that other word processors and editors can also read it. When you send such a file to another machine, perhaps one of the IBM PC compatible machines running PC or MS DOS (Disk Operating System, the PC's internal traffic controller, similar to Macintosh's Finder), it can use the file with its own programs.

Tabs The Macintosh displays most of its fonts proportionally: each character takes up the amount of space on the line *proportionate* to its shape. This makes it easier to read, aesthetically more pleasing, and gives the Macintosh the ability to display different kinds of typefaces.

Fixed character spacing, on the other hand, is what most typewriters, a lot of computer printers, and most non-Macintosh computer display screens do: each character takes up the same amount of space regardless of whether it's an *M*, an *m*, or an *i*. This is also the way text is presented on most terminal screens. The IBM AT is typical: text is displayed with 80 characters across and 24 lines down. This is sometimes called a *fixed-pitch font display*.

For information presented in tabular form, such as an excerpt from a spreadsheet or other kinds of tabular data, the difference between a proportional font display and a fixed-space display becomes a special problem. Most word processors on non-Macintosh machines measure a fixed number of character spaces from the start of a line to determine where a tab stop is placed. The Macintosh, on the other hand, positions its margins and tabs on a ruler. A skip to a tab stop, then, is measured as a fixed number of inches from the edge of the display, regardless of how many characters preceded the tab.

When going from a non-Macintosh to a Macintosh environment, the appearance of documents that used tabs and documents that used "hard spaces" to align text material within documents will show up on the Macintosh looking quite different. Conversely, when going from the Macintosh to another machine, documents that have tabs in them (represented by ASCII 9) will also arrive looking worse for the trip.

The best solution is to transfer material between the same application running on the two machines. For example, Microsoft Word runs on both the Macintosh and the IBM PC family.

In some cases, you can apply a translation program that is smart enough to translate from Macintosh format to non-Macintosh format and back again. For example, in our model Macintosh-to-PC system in Chapter Six, we recommend use of MacLink. MacLink translates the

ruler information in Word documents into the formatting codes required by MultiMate. This is quick, easy, and painless because the translation occurs during the actual file transfer. In the worst instance, you will have to realign the text by hand at the keyboard.

Formatting ASCII is a seven-bit code. All information on disk is stored in eight-bit chunks called *bytes*. In a vanilla text file (unformatted ASCII text characters) the eighth bit is set to zero, but from the disk's point of view, it's still there.

Most good word processing programs give you this option of saving text in a vanilla ASCII text file. Such files can then be transferred to another computer—just about any computer, in fact—using the Send Text option (or its equally prevalent equivalent on other terminal packages).

Because of its generic and relatively easy implementation, the exchange of ASCII files has become a nearly universal function. Of course, there are drawbacks to this method, which is why other techniques have been created. A drawback with saving and sending textual information this way is that word processor formatting is lost in the transfer. You would then have to “tweak” the received ASCII file; that is, go into the file with your own word processor and put back all the formatting information lost in the transfer.

This problem of different formats (or lack of standards, if you prefer) comes up over and over again when attempting to exchange information electronically, not just with text files. On the bright side, many bridges have been built between most of the major word processing, data base, and spreadsheet packages. On the darker side, the ability to exchange graphic information between different programs on different machines is nowhere near as developed.

If the majority of your documents are short memos and manuscripts, you might not mind losing file formatting, and simple ASCII transfers will do the job. However, if you need to exchange formatted information regularly, or graphic information, you may need to look more closely at the task.

One of the first places to look is at the various applications programs themselves. Many of them provide built-in translators or special format *bridges* you can use to massage a file before sending it to another system.

For example, Microsoft developed the SYLK format to facilitate exchange of information between its spreadsheet packages (Excel, MultiPlan) and other programs. Similarly, Software Arts (creators of

the original VisiCalc spreadsheet) developed DIF (Data Interchange Format) for the same purpose. Lotus development uses a format distinguished in the PC DOS world by the .wks or .wk1 file extensions. These files can be read by 1-2-3, and by other programs, such as Ansa Software's Paradox data base for the IBM PC family of micros.

So, before assuming you can't exchange a file with a colleague, check to see what kinds of file formats can actually be used or created by his or her own application. By going through an intermediate translation provided by the application software on either end, you may be able to share the data after all. We cover this ground more thoroughly in the Macintosh-to-PC section of Chapter Six.

Yet another drawback to ASCII-oriented file transfers is that non-text information such as MacPaint documents cannot be sent. Nontext files use all eight bits of every byte to hold vital information. The eighth bit is lost if ASCII text file transfer is specified. It gets dropped, ignored, stripped, or otherwise changed as it goes through the transfer process.

If the contents of a file consist of text material or alphanumeric information (text and numbers) that is not error-sensitive, and if the file will be used at the other end by an incompatible word processor, text-only transfers are the quick and easy way. But what is alphanumeric information that is error-sensitive? An example might be financial data being transferred for processing between offices. You wouldn't want your paycheck to reflect payment of 34.5 hours when what you actually reported was 44.5. The 4 may have been changed to a 3 as the result of a text-only file transfer on a momentarily noisy phone line.

A non-error-sensitive file might be a memo or document in which the equivalent of a few typographical errors will not substantially alter the meaning or the consequences of the text. Text errors in the form of typos are easy to spot.

Another example of an error-sensitive file is covered in the slightly more complex case of transparent file transfers.

Transparent File Transfers

Suppose you and your associate both have Macintoshes. Your task is to exchange with one another MacPaint documents containing the storyboard information for a new videotape series the two of you are producing.

If you both worked in the same building, the problem would be pretty easily solved; you could just exchange disks with the MacPaint documents on them. If you were going to exchange information on a daily basis, a direct-cable link and a program like MicroPhone would

allow you to ship files back and forth quickly without having to exchange disks back and forth. With a program such as Hayes' Smartcom II, you could even interactively codesign your drawings.

However, our picture gets complicated if your two machines are in different parts of a city or across the country. If you mix in two full schedules, and neither of you want the hassle of coordinating a connection, what do you do?

You use an intermediate machine to hold your MacPaint files until they can be picked up by the other party at his or her convenience. (Any system will do: a public information service, such as MCI, CompuServe, or Delphi; a private BBS that supports the appropriate file transfer protocol; or any other system that's available.)

The main criterion in this kind of transfer is that the whole process of moving the information must be totally transparent, from beginning to end. A communication channel is said to be *eight-bit transparent* or just *transparent* when all of the bits in a file maintain their relative values and positions in the file.

Macintosh graphic information is stored in binary form (read about binary files in Chapter Eight). You can still transfer and use such files, but you can't use an ordinary ASCII text type of transfer.

In order to accomplish a full eight-bit transparent file transfer, our communications software, through whatever error-checking file transfer protocols it supports, must be able to eliminate the possibility of error in even a single bit during the transfer. If we want to combine file translation with bit transparency, we have an even more demanding task to perform.

This is where our other two Send options on the **FILE TRANSFER** menu come in. MicroPhone employs several error-checking file transfer methods. The MacBinary and MacTerminal protocols consolidate and preserve both forks of a Macintosh file as it is being transmitted. Xmodem alone will transfer the data fork of a file, complete with error checking. The Xmodem kind of exchange predates MacBinary, which was designed specifically for use with Macintosh files.

Both MacBinary and Xmodem, as well as additional methods such as BLAST and Kermit (primarily for mainframe and transnational communications), are covered in greater detail in Chapter Six and in the advanced technical topics of Chapter Eight.

File Transfer Settings Revisited

Earlier in this chapter when we were looking at MicroPhone's File Transfer Settings dialog box, we promised that we'd return to the last

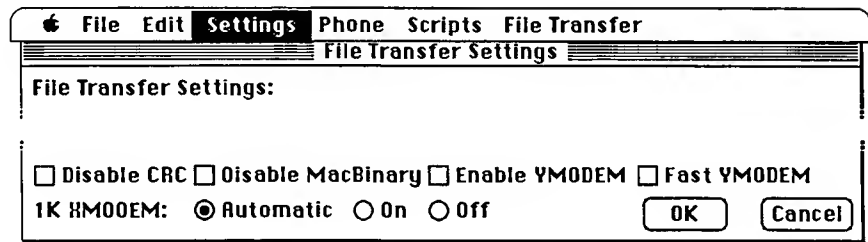


Figure 4.19 File Transfer Settings Revisited

two lines of the box (Figure 4.19) when we discussed the **FILE TRANSFER** menu selection. That time has come.

These selections give you additional control over the MicroPhone's Xmodem file transfer protocol. In practice, you should always try to use an eight-bit transparent protocol when communicating Macintosh-to-Macintosh even for vanilla text files. Microphone's Automatic selection will accommodate itself to either 1K block or 128 byte block transfers. You can send text only by using that option from the **FILE TRANSFER** pull-down menu.

Not all systems will support the same set of eight-bit protocols. In fact different versions of the "same" protocol may not communicate with each other. Right now, if you acquire a program that supports Xmodem, MacBinary, and Kermit, you'll be able to accomplish just about every communications job you want to do with your Macintosh.

That brings us to the end of our tour of MicroPhone. We have covered all the major elements of any Macintosh file transfer using telecom.

You'll find a telecom software feature comparison chart at the end of Chapter Six. That chart mirrors most of the major communication software elements we covered in this overview.

Now, in the remaining chapters of *MacAccess*, we'll explore more ways for you to get high performance out of your telecom equipment, and equip you with information you'll need to deal with any communications problem that might arise.

CHAPTER

5

Communication Command Languages

Necessity never made a good bargain.

Franklin

Too much of a good thing is wonderful!

Mae West

It's easier than ever for Macintosh users to access the benefits of data telecommunication. There are now many packages from which to choose when shopping for an all-purpose terminal for the Macintosh. Even if your need is specialized, such as turning your Macintosh into a Tektronix-compatible graphics terminal, you can probably find an off-the-shelf solution.

If you're looking for even more power than your current terminal package gives you (maybe you're still using MacTerminal 1.1), or if you want to make access even easier for others, a fine crop of programmable terminal packages is waiting for you. *Programmable* means that these products include a high-level, telecommunications-oriented programming language along with the basic functions we've come to expect of terminal software. These programming languages offer an easy path to advanced automation for those of us whose telecom demands go beyond the basics. The programming features remain in the background until they are needed. The novice doesn't need to know anything about the programming capability of the software because all the basics are presented through the Macintosh desktop, menus, and icons.

We're going to look at four different communications software packages that feature programmability as part of their user appeal: inTalk v.2.1 (Palantir), MicroPhone v.1.035 (Software Ventures), Smartcom II v.2.2 (Hayes), and Red Ryder v.9.2 (Freesoft).

Perhaps in an effort to make their programming languages sound friendlier and easier, or perhaps to position themselves apart from one another, or both, different companies call their programming capability by different names.

- Hayes calls its Smartcom II programming utilities Autopilot.
- MicroPhone uses the term *scripts*, and a script-recording feature called Watch Me. MicroPhone will record your interactions and write a script for you automatically with the Watch Me feature.
- Red Ryder uses Remote Service Procedures or just *procedures*. Red Ryder records your interactions and writes procedures with Write a Procedure For Me.
- Finally, inTalk has a CCL, which is the generic and most accurate name for such built-in languages. It's also short and easy to remember and type.

We prefer the terms *scripts* and *programs* to refer to the written commands used to program a telecom sequence. Here, we'll use script or program to mean a MicroPhone script, a Red Ryder procedure, or a block of code written in inTalk's Communications Command Language.

If you find yourself connecting with a service or an associate's micro on a daily basis, collecting new messages and sending responses, you can use scripts to program your Macintosh so that it can attend to these matters for you, unsupervised. A telecom command language will also let you set up automated systems for clients or coworkers, so that they'll require even less training in telecom. Small work groups can be linked together in customized, tailor-made ways.



Time Invested vs. Degree of Automation

Legend tells of a great Navy man, an Admiral, who got to the top by leaving every job he'd ever had in a simpler yet more efficient state than he found it. He claimed it was because he was basically lazy. His laziness took him to the top just as surely as ambition.

His sequence of rapid promotions followed a pattern. Whenever he encountered a new set of responsibilities, he went about learning all he could about the situation. As he learned, he stayed alert for ways to simplify matters so that they took as little supervision from him as possible. Most of the time, he delegated decision making and routine items to subordinates. (This trick is well known in management schools.) Sometimes, though, he would have to invent a new procedure or a new way of accomplishing a task.

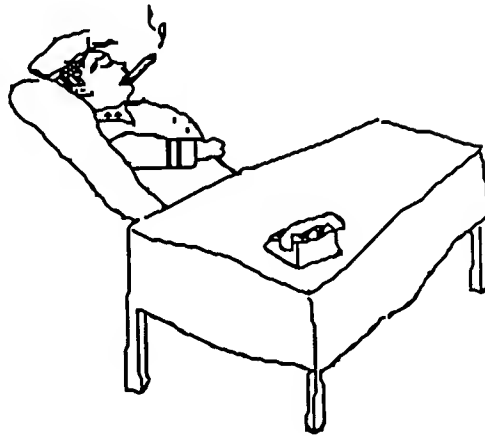


Figure 5.1 Lazy Admiral

As he implemented his new policies, things tended to function more smoothly, and by the time the authorities got the results (in terms of increased productivity figures, less wasted overhead, and the rest of the accountant's numbers) he had already been able to enjoy the fruits of his efforts by a few extra weeks or months of a relatively lighter workload. As soon as his superiors caught on to how well things were running, they would promote our man again, and the cycle repeated itself.

This "more with less" syndrome is well known in the software field. From the days when switches representing zeros and ones had to be manually set in order to "program" a calculating machine, to today's full-bodied applications and high-level programming languages, thousands of person-months have been spent to reduce human effort by delegating more and more of the routine things to the machines. It's a good thing, too, because we have so much more to try to do less with these days.

This situation is a wonderful paradox: freedom and individuality tend to render everything more complex, but we can employ our own freedom and individuality to simplify matters for ourselves if we wish.

For purposes of comparison, the chart in Figure 5.2 shows the relationship between various degrees of telecom automation and the up-front time required to make the automation work. For example,

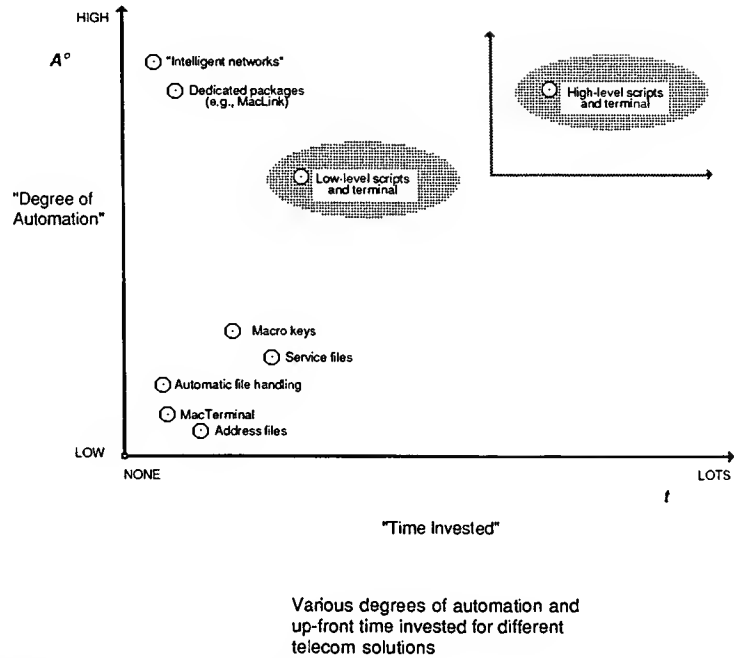
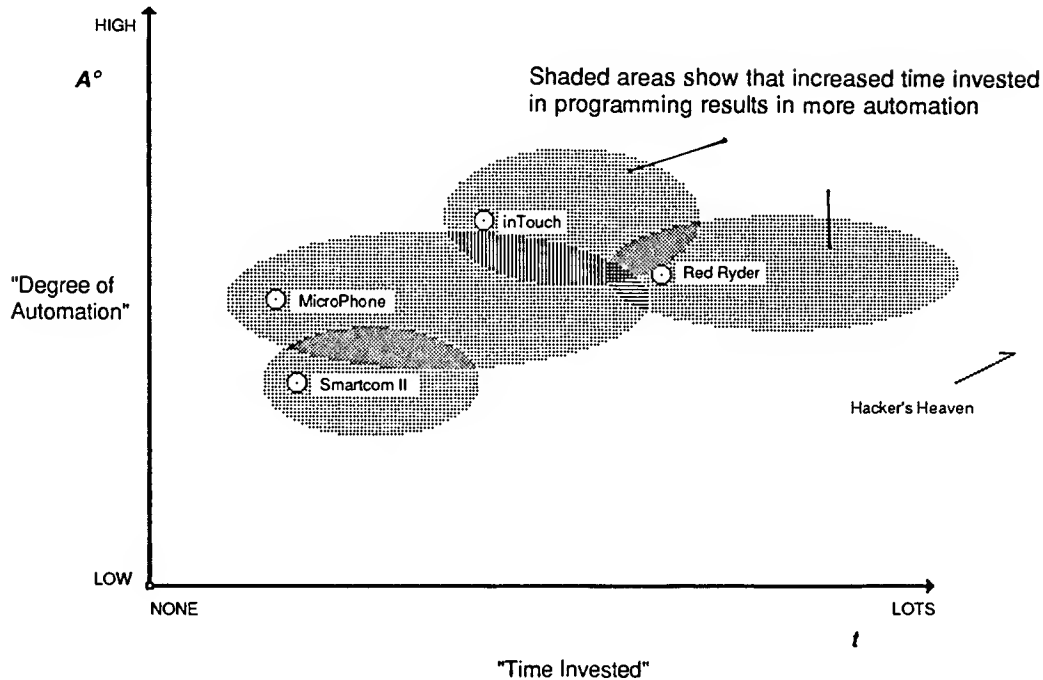


Figure 5.2 Automation vs. Time

such an investment would include time spent installing a package and entering your own phone numbers and communications parameters. In the case of packages that have macro capabilities, the time you spend programming a function key or “mousable” button with your own commands is up-front time that must be invested before any benefit of automation can accrue.

The points on the chart show some of the generic automation solutions available to users of telecom software and services. Note we aren’t dealing with dollar costs here when we talk about investment, only time. Note, also, that two of the points are surrounded by gray, which is meant to show that by investing more time, you can gain additional automation benefits; the chart really represents a *range* of time versus benefit tradeoffs.

Terminal programs that incorporate high-level command languages occupy the upper-right quadrant which we show “expanded” in Figure 5.3. As implied, the time you must invest in mastering one or



Close-up of terminal programs with
high-level command languages

Figure 5.3 Scripts Compared

more of these languages is amply rewarded, in our estimation, by the degree of automation that can be obtained.

Figure 5.3 shows the relative positions of the four terminal packages we've chosen to look at here: inTalk, MicroPhone, Red Ryder, and Smartcom. Again, this impressionistic graph is meant to provide a visual aid for comparison purposes. The shaded areas show the range of possibilities for each package. Note, too, that with additional work and a lot of cleverness, even the less-powerful packages can be made to perform wonders in a particular situation.

Before we get to the different features found in each of the script languages provided by each of these particular packages, let's step back for a moment and look at the levels of possibility covered by Figure 5.2.



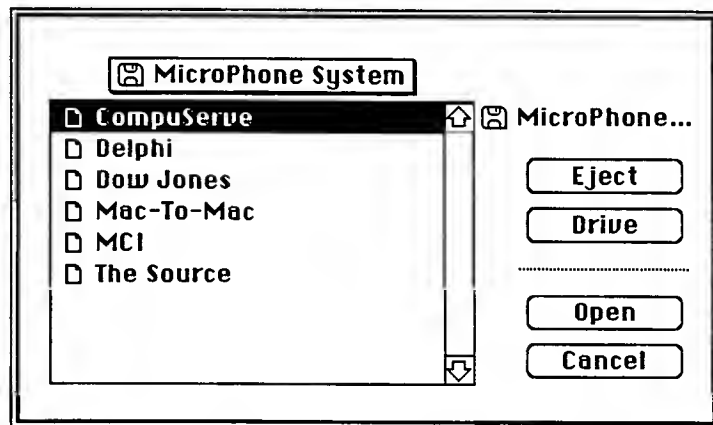
Low-Level Automation

It is rare to find a general-purpose communications program (terminal emulator) that does not include all of these functions in its basic plan: service files, macros, and automatic file transfers.

Service or Address Files

A good terminal package will have a phone number maintenance utility of some kind. This built-in address file management is frequently combined with automatic dialing. You simply select the name of a remote computer service from the list presented (Figure 5.4), and the terminal (Macintosh plus software) dials and connects with the other system automatically. Your time investment includes the time for the initial entry and file setup plus whatever update time may be required after that. This is really a base-line investment for all the packages we cover here, since without this feature, you would have to enter the phone number and settings information each time you wanted to link up. That wouldn't be convenient at all.

Service files store the communication parameters needed for the particular service (bit word length, parity, baud, and so on), and any required flow control settings (X-On/X-Off, line lengths, and so on).



The MicroPhone disk came with these services. The telephone number, and communication and terminal setting (and scripts!) are retrieved when you open the file.

Figure 5.4 MicroPhone Directory

Macros

The next feature found on most terminal programs is the ability to create macro commands like the ones shown in Figure 5.5. A *macro* is a sequence of commands (or keystrokes), stored as a command string. Sequences of commands and remote interactions that are repeated over and over can then be performed with one or two actual keystrokes once the macro has been set up. By creating *macro keys* for each system you connect with, you can cut down on your on-line time. As an added advantage if you're not that good a typist, you can cut down on retyping time, too. Macros and macro keys are a feature of many different kinds of software, not just telecom packages.

Macros are quite useful while on line. For example, you can assign a log-on sequence or password to a macro key (or keys). Then, when you're prompted by the other system, you press the macro key and the whole string of characters is sent for you.

Some packages limit the number of macros you can have available at any one time. Others allow you to call macros from inside other macros. Still others, while limiting the number and lengths of the

Macro name	Instructions for actions to be performed
Logon/ Logoff	^E "CSI Logon"
Mail	GO EASY^M^\$L2
Quotes	GO QUOTES^M^\$L3
News	GO NEWS^M^\$E "CSI News"
Travel	GO TRAVEL^M^\$E "CSI Travel"
Avia- tion	GO AVIATION^M^\$E "CSI Aviation"
Mac Users	GO MACUS^M^\$L4
Setup Vidtex	^\$E "CSI Vidtex"

Level 1 Level 2 Level 3 Level 4 OK CANCEL

inTalk lets you define up to 32 macros
(4 sets, or "levels", of 8 macros).

Figure 5.5 inTalk Macros

macros. Still others, while limiting the number and lengths of the macros available, will allow you to get around this limitation by providing the ability to swap macros in and out of memory as needed. This approach begins to look like programming. Programs are written using the command strings provided for by the macro language. Here's a macro sequence written in a typical macro-string language. See if you can tell what the macro does by reading it.

```
• M@W • M@E@T=d1 • M@T@C 30128 • M@T>ID STR555 ABCDEF • M
```

Don't worry. We couldn't tell what it does either, without looking it up in what passed for documentation. After a painstaking deciphering, we found out that this mysterious string of characters automatically establishes a connection with another computer and then sends out an identification number.

As you can see, macros may be a step forward in automation, but they are sometimes a step backward in understandability. With a little effort, though, they can at least give you some relief from the dialing drudgeries.

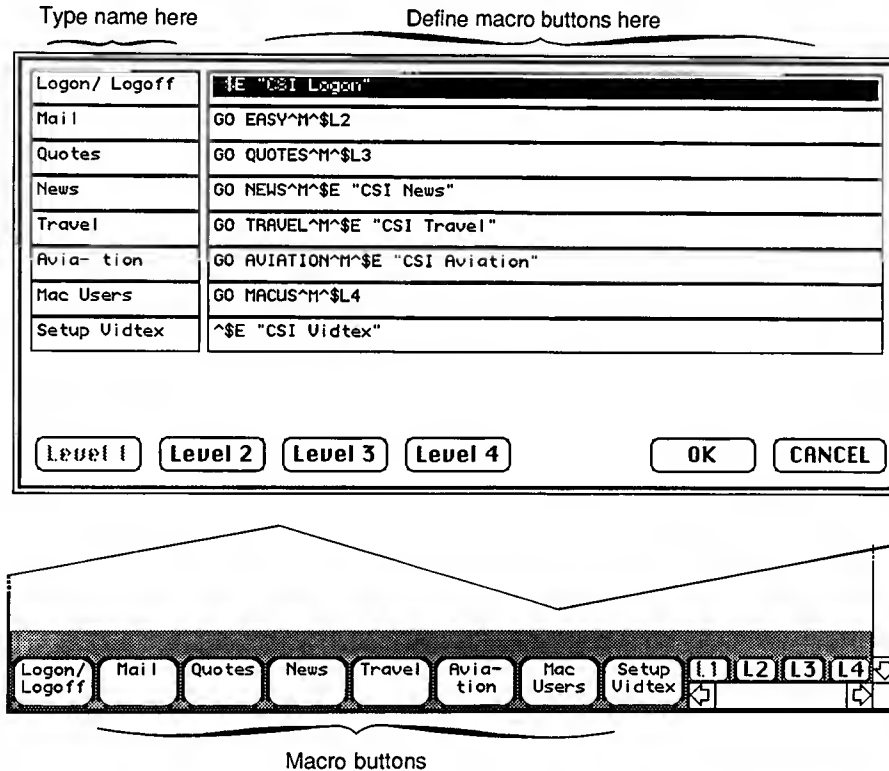
Once they are created, there are three common ways to get to the macros and execute them using the tools provided by the Macintosh interface:

- They can be selected from pull-down menus.
- They can be called through a command-key sequence.
- They can be mouse-clicked from on-screen buttons and boxes.

If you have a preference for any of these ways of interacting with your Macintosh, then you might want to look closely at how a given program implements its macros. Our own preference is for displaying macro buttons or menu items on the screen, as shown in Figure 5.6, where they can be selected by mouse point-and-click actions. This saves time because you don't have to memorize command keystrokes.

The difference between a macro command set and a high-level programming language is more than a matter of degree. Macros are usually limited in terms of the number of commands you can string together at once. They also may not allow testing for conditions to direct program flow.

Some macro command sets let you write a macro sequence that will wait and test the incoming bit stream for a set response before



inTalk lets you run macros quickly by clicking the appropriate button.

Figure 5.6 inTalk Buttons

continuing. For example, you might tell it to look for the character string COMMAND and then send a carriage return. This gives you a degree of control over how the macro interacts with the remote system but not as much as a script gives you. A full-blown script language will not only let you watch incoming characters for a set response before continuing, but it will also be able to select alternative paths based on the response. Finally, script languages are generally closer to English, with command names that give you some idea of what the command does, such as Send File rather than ^S ^F.



Up Tempo

We've used it and we're hooked. Tempo is a piece of Macintosh ingenuity disguised as a desktop accessory. Tempo lets you record sequences of mouse-and-keyboard actions and play them back at any time: instant macros! Sequences recorded while working with an application (Red Ryder, for instance) can only be played back within that application. This means that it doesn't matter any more, what program you're running, you can have a set of macros that conform to the way *you* do your work with any program you use. So, if you find a public domain or commercial terminal program that you like, but one that doesn't include macros, Tempo will fill the gap. (MacTerminal 2.0 users, take note!)

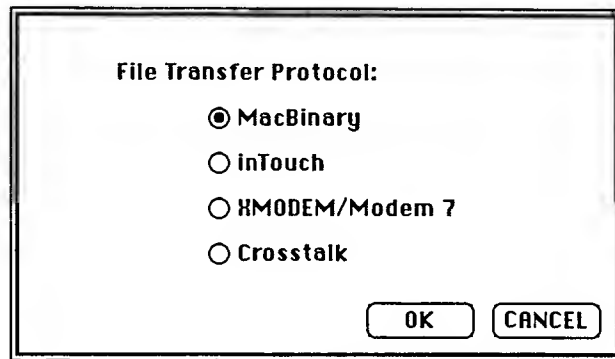


Behind the Scenes Conversions

This bit of automation removes a whole layer of concern from the minds of Macintosh users. Terminal programs that perform automatic file conversion from one format or disk storage scheme to another during the process of transfer make housekeeping and file-preparation chores a whole lot easier. Consider this: to transfer a Macintosh program file (application) to another Macintosh through an intermediary system without using MacBinary (the Macintosh-to-Macintosh file exchange protocol) requires a conversion using another utility called BinHex. BinHex is used at both ends of the transfer. It's used once to get a file into a form that can be easily sent and a second time to reconvert it back to a form useable by the Macintosh. Red Ryder, by the way, eliminates one of those steps by automatically converting BinHex files as it receives them. By using MacBinary (Figure 5.7), the transfer is made directly (provided the other system also supports MacBinary) with the transfer protocol, in effect, taking care of all the intermediate file-messaging tasks.

Putting It All Together

By combining sophisticated telephone management support, an expanded macro ability, and automatic file-manipulating abilities with the additional control features found in high-level script languages, you get a terminal program like one of those shown in Figure 5.3. Each of these packages provides built-in programming. Each combination of features for the main program (all those functions you can get to and use without having to know anything about programming) and for the



inTalk lets you choose between four protocols

Figure 5.7 inTalk MacBinary

scripting language are slightly different from one to the next. Each qualifies as a good, general-purpose communications program. Each is appropriate for different kinds of use.



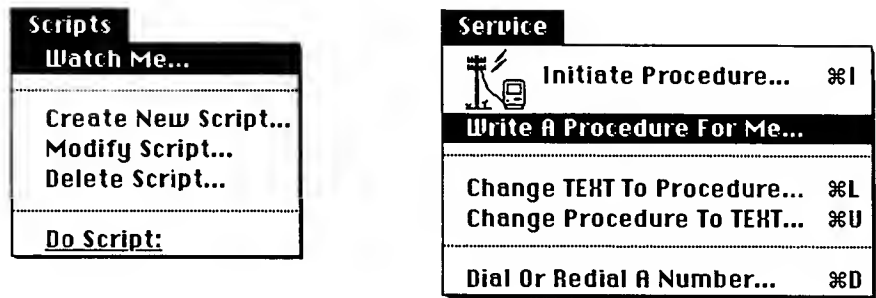
Our Ideal Script Language

What are the features to look for in an ideal telecom-oriented programming language? That depends on who you are. If you are a complete novice, your ideal package might require something like the ones shown in Figure 5.8.: MicroPhone's Watch Me feature, or Red Ryder's Write a Procedure For Me. These options can be thought of as built-in program recorders that capture your session keystrokes and the remote system's response for later playback. In effect, this automates the process of creating lengthy sequences of commands: automated automation!

If, on the other hand, you are an experienced programmer or professional telecom consultant, or if you customize your communications with each of your clients, you will need to be able to use a rich set of control structures.

In between, there will be individual "must have" and "can do without" lists. What do we expect our full-featured and robust language to do?

- Perhaps foremost, it has to provide a means of controlling the telephone line and modem. The script should have commands that answer and dial the line. This implies, of course, that you have a modem that is capable of answering and dialing out, commonly called autoanswer/autodial modems. No soft-

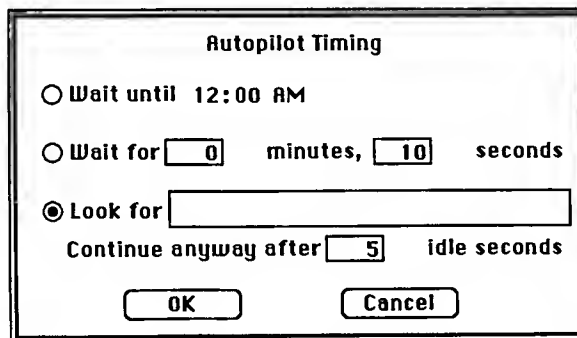


MicroPhone (left) and Red Ryder (right)
will write scripts for you

Figure 5.8 Watch Me/Write Me . . . Make Me Do Fast Scripts

space should be without one, and the rest of our discussion of scripts assumes autoanswer/autodial modems.

- It should also have a time control function like the one shown in Figure 5.9, so that the program will wait until a specified time before dialing or setting itself up to answer the phone.
- There should be a facility for automatically redialing a number (attack dialing). You should be able to specify the number of trials and the amount of elapsed time between trials. Figure 5.10 shows inTalk's attack dialing feature.
- If it can answer the phone, it should also have the ability to provide the dialer with some access to your Macintosh files. Therefore, some ability to password-protect on-line access is helpful. Also, the dial-in user should have complete control over the connection so that your Macintosh can handle the transaction unattended, if you desire. You should be able to block access to selected files. Some terminal packages include the ability to autoanswer the telephone, but they require a person to be on hand to supervise what happens next.
- Along with phone and password management, the ability to automatically log calls (in and out), record on-line connect time totals, and make note of unsuccessful tries can give you additional valuable control.
- The script language should feature commands that watch the incoming stream of characters for specific string-events: user prompts, command requests, and so on. The language should have conditional and unconditional branching options like the ones shown in Figure 5.11 so that different program flow paths can be taken depending on the conditions.



Autopilot Timing

☐ Wait until 12:00 AM

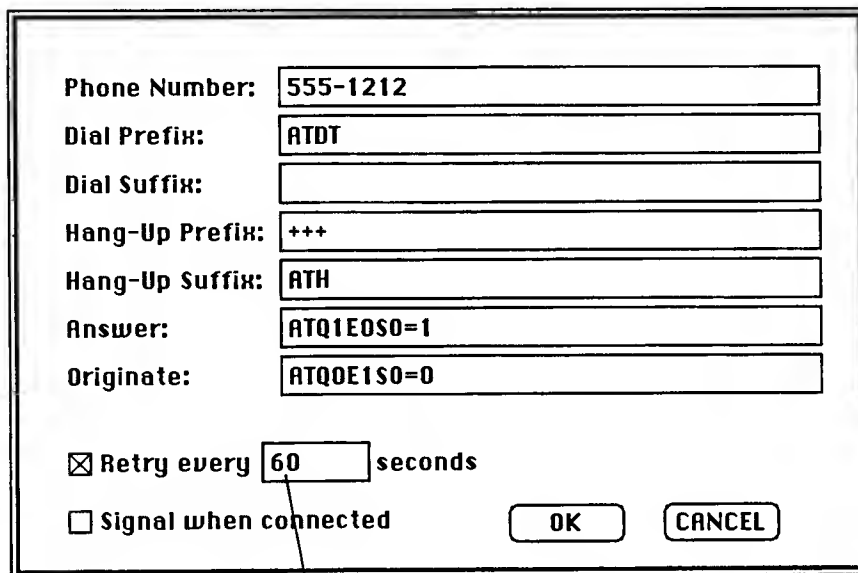
☐ Wait for minutes, seconds

☒ Look for

Continue anyway after idle seconds

Smartcom's Autopilot will wait for a certain period or until an appointed time before continuing its actions

Figure 5.9 Smartcom Timing Control



Phone Number:

Dial Prefix:

Dial Suffix:

Hang-Up Prefix:

Hang-Up Suffix:

Answer:

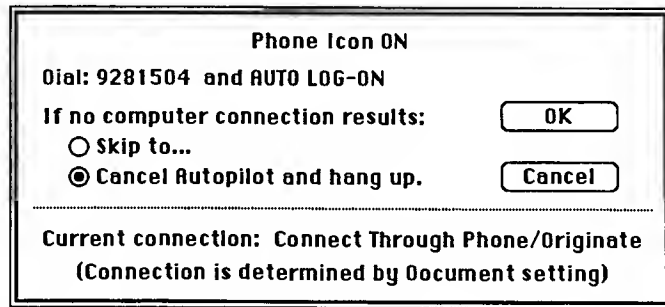
Originate:

☒ Retry every seconds

☐ Signal when connected

inTalk will redial a busy number at a specified interval

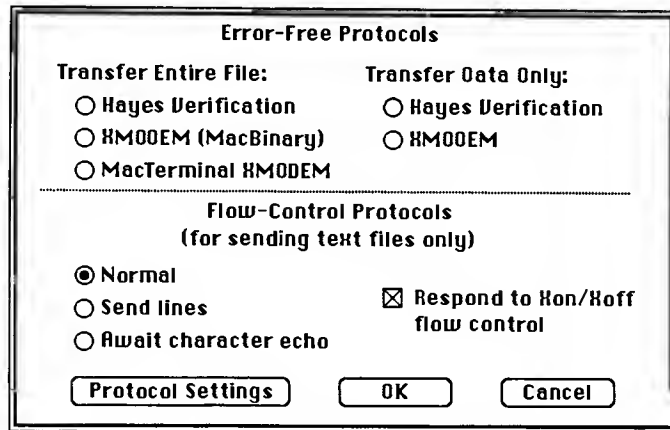
Figure 5.10 inTalk Attack Dial



Smartcom branching boxes lets you make decisions about what to do when certain events occur

Figure 5.11 Smartcom Branching

- There should be one or more functions that are able to get input strings from the local user, if necessary. There should be some means of storing these responses, which usually means being able to define either *numeric* or *string* variables by name or by number, preferably by name.
- Error control should be provided that is capable of handling the most common communication problems without a glitch: noisy lines, interrupted (incomplete) transfers, excessive line delays, and slow response times are the most important to plan for. Some of this control is dependent on the kinds of transfer protocols being used or supported. Protocol error checking and facilities to monitor results within the program (Figure 5.12) go hand in hand.
- Errors arising from the wrong or unexpected input from the other system should be easily trapped and sidestepped by the script. An executing program should never cause the system to freeze up altogether, arbitrarily disconnect the line, or otherwise perform in such a way as to result in being connected to another system for long periods of inactive time. Getting it to perform properly is partially a function of the commands available, and partially a function of how you use the commands.
- Like the one shown in Figure 5.13, a language should include the ability to store comments within the program so that its functions can be annotated for later change.
- You should be able to edit script files with either a built-in editor or your own word processor—preferably both.
- There should be some sort of debugging facility like the one in Figure 5.14. This means, at the very least, that the system should provide you with information about where a program stops or encounters errors.



Smartcom's offering of transfer protocols

Figure 5.12 Smartcom Protocols

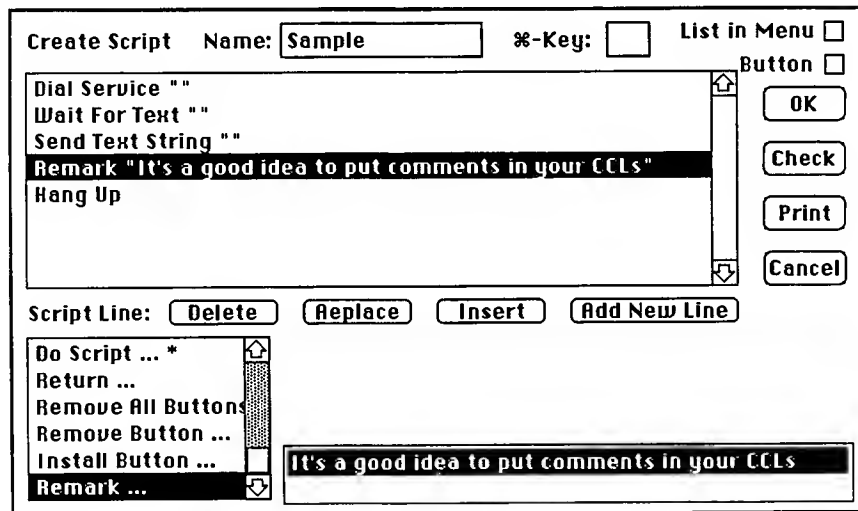


Figure 5.13 MicroPhone Remark

These features will vary in importance from situation to situation. Individual languages present different combinations of these features, and none have all of them. If you are choosing a terminal package on the basis of the quality of its programming features, which one you end up with will depend on your needs and preferences. In order to give you a feel for the capabilities of different script languages, we created the following short sample script.

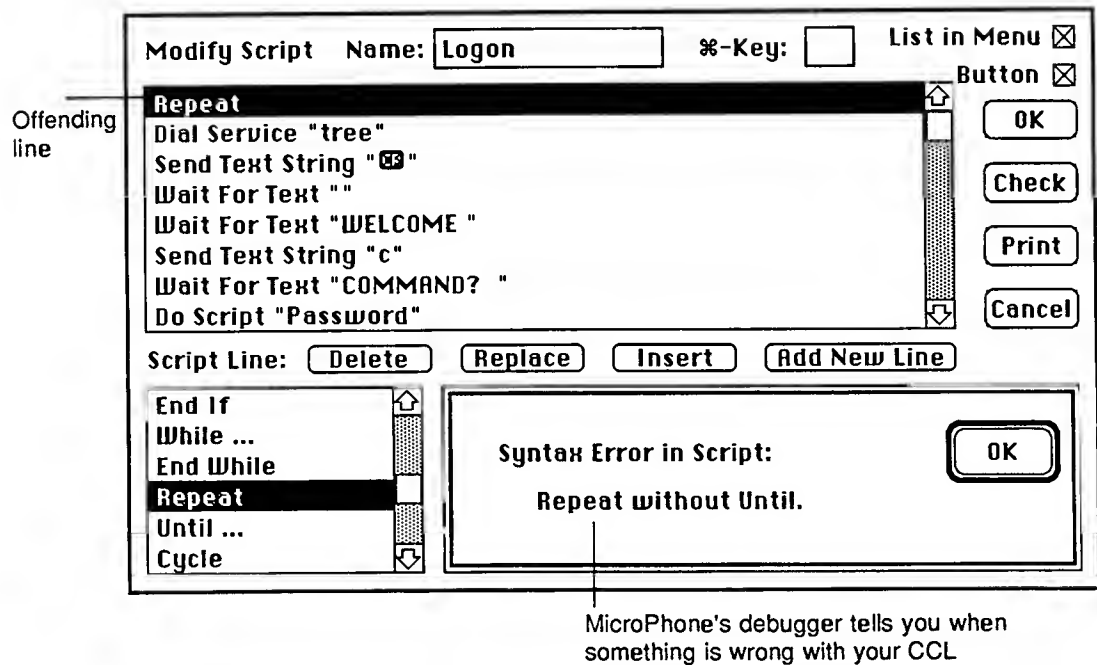


Figure 5.14 MicroPhone Debugger

Sample Program

Imagine this scenario: A workteam from UpStart software has installed a Conference Tree (Appendix C) to support their design efforts. Each member of the work team uses a slightly different combination of hardware and terminal software. Two members of the team use Macintoshes, and they want a program to automate their communication. Our task is to write the Macintosh program that does just that.

We used the Conference Tree to develop our example programs partially because the Tree is consistent in its log-on procedure from session to session. Some host systems will vary their response depending on their knowledge of the user: the kind of terminal, job function, and modem speed, whether or not there is electronic mail waiting, whether the system operators want to grab your attention, and so forth. Simple programs may break down whenever the host does something not anticipated by the program, that is, something for which there is no prepared response. Dealing with such inconsistent hosts requires a lot of programming cleverness and debugging time.

In developing this sample, we'll use the more-or-less traditional, top-down design approach. First, we describe the application in everyday language.

Background and Description of Program Function The Macintosh script (program) should automatically dial the other system (Tree), logon, and present the system password. The program then provides options for automatically performing the following:

- Getting new messages put on the system since their last session
- Adding messages to the system
- Quitting (logging off)

The next step is to take our general description and turn it into so-called pseudocode, a more structured English representation of the program's flow. Here is the pseudocode for the program Tree Logon.

Logon

Dial number

Connect and start session timer

Display first screen

Issue password

Display additional on-line buttons for

- READ new message
- ADD new message
- DELETE message
- Send "Conferences" string
- QUIT

Finally, from the pseudocode, the actual program is written. Figures 5.15 and 5.16 show the same functions as represented in a MicroPhone script and inTalk program, respectively.

Note: The terms CONFERENCES, ADDTO, READ, and QUIT are words that control the Tree: they are Tree-specific commands.

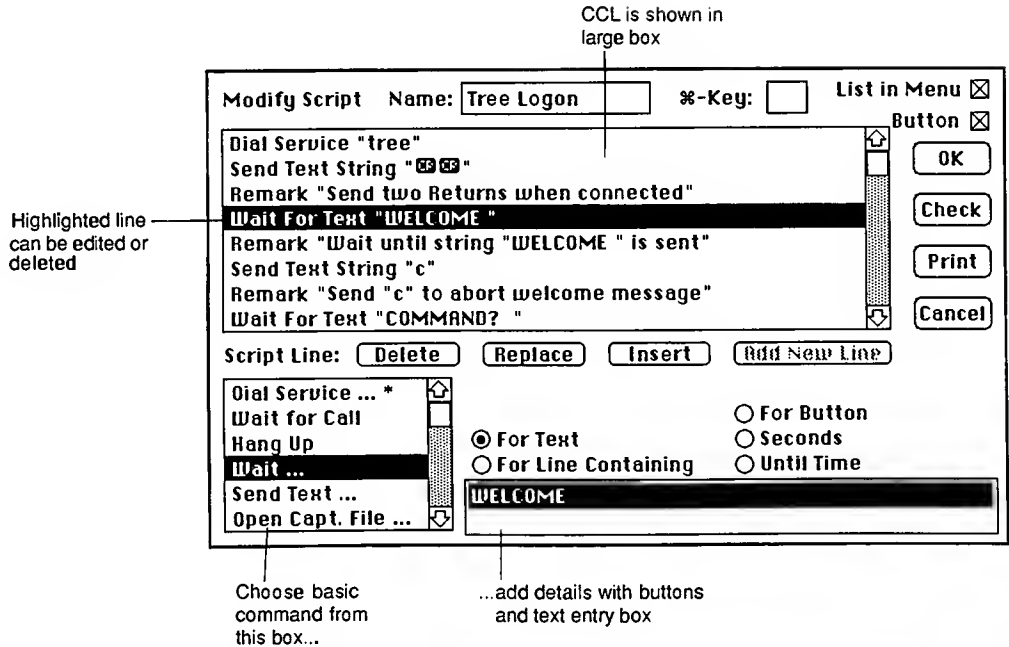


Figure 5.15 MicroPhone Script

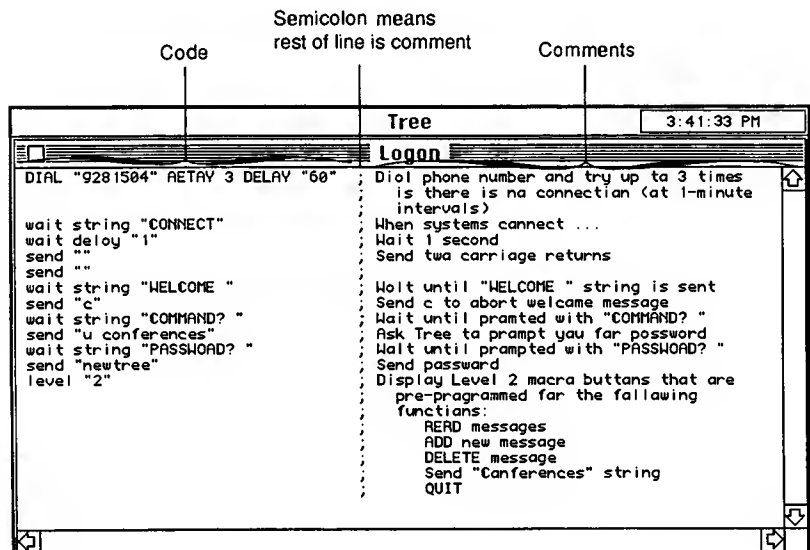


Figure 5.16 inTalk Script

Here's the full text of the MicroPhone script:

```
Dial Service "tree"
Send Text String "<CR><CR>"
Remark "Send two returns when connected"
Wait For Text "WELCOME"
Remark "Wait until string "WELCOME" is sent"
Send Text String "c"
Remark "Send "c" to abort welcome message"
Wait For Text "COMMAND?"
Remark "Wait for 'COMMAND?' prompt"
Send Text String "u conferences <CR>"
Remark "Ask tree to prompt you for password"
Wait For Text "PASSWORD?"
Remark "Wait for 'PASSWORD?' prompt"
Send Text String "newtree<CR>"
Remark "Send password"
Remove All Buttons
Remark "Clear out old buttons..."
Remark "... and add Conferences, GetMail, AddMessage and QUIT buttons"
Install Button "Conferences"
Install Button "GetMail"
Install Button "AddMessage"
Install Button "QUIT"
```



Tech Tip

When MicroPhone asks if you want to save the settings, say "yes" if you want to save changes you've made on a script. In this case, settings will include the script or scripts associated with whatever MicroPhone service file you have open at the time.

Once the code for a program has been composed, it must then be tested and debugged using the actual services for which the programs are intended. This part requires some patience because you are dealing with the quirks of the remote system and the behavior of the script language being used as they interact with one another. There is no substitute for actual experience at this stage of the process.

Discussion and Comparison

MicroPhone

Even for beginners, MicroPhone's script language is easy to read and understand without extensive internal documentation in the form of remarks.

A short script is equivalent to a macro, which can be accessed through either menus or on-screen buttons and selected with the mouse. In our sample program the various functions we wanted to include (such as QUIT) were broken up into conceptually complete modules and stored as their own scripts with their own names. You can branch to these scripts from within a given program and return to the original program again after the secondary script has been performed.

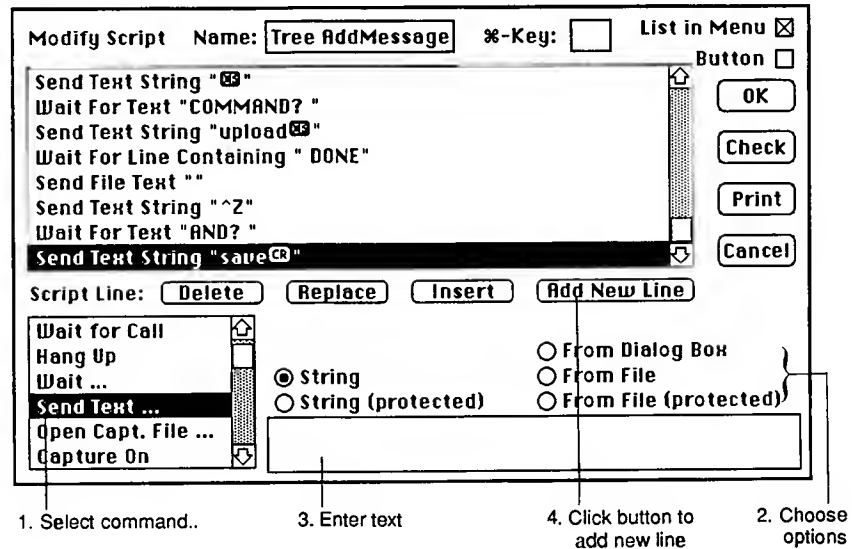
You must use MicroPhone's built-in script editor to create scripts. Once they are written, they may only be modified through MicroPhone, MacWrite, or any other word processor, won't touch them. Figure 5.16 shows the script-editing screen with our commentary. Experienced programmers might find that the hand-holding provided by this editing dialog box to be a bit tedious, but we found that being able to write a program using little more than the mouse and occasional keyboarding more than make up for MicroPhone's overly supportive features. You can't turn off this support, and we found that we could live with it. Programming MicroPhone is more like using an *application generator* (Figure 5.17) that writes the actual computer code for you while you make selections and choices from dialog boxes.

With MicroPhone's Watch Me menu selection (Figure 5.18), you can begin recording an on-line session at any time and capture the sequence of commands exactly as you encounter them. This feature lets you skip the process of writing a top-down design document or pseudocode. Writing a script then becomes a matter of recording the sessions that do what you want to do and then linking the sessions together into modules or scripts that are *nested* (one inside the other).

Watch Me scripts can seldom be used "as is." Once it's been recorded, more than likely you'll have to do additional work to edit it and put in contingencies for error handling.

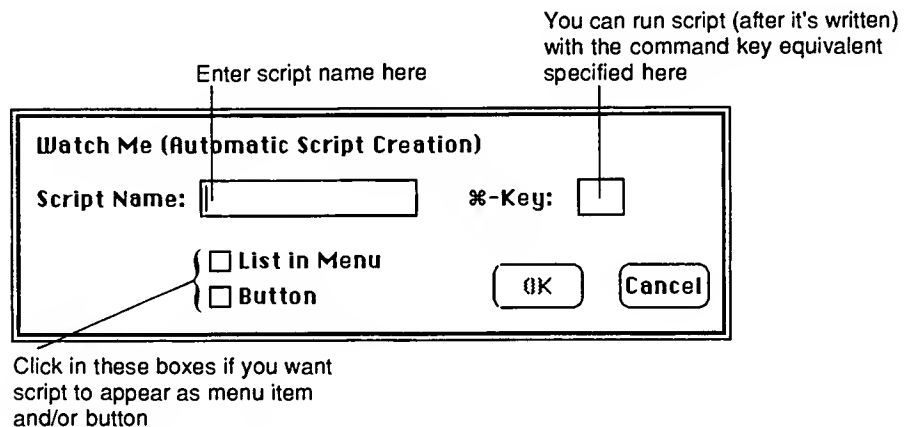
By clicking Check in the editing dialog box after you've completed a script, MicroPhone will go over the lines of code and tell you if there are any obvious errors, such as misspellings, syntax errors, or parts of commands left out. This is a first-level debugging facility. It can't find

errors in text strings that you expect from the remote system, of course. For that, you have to get on line and actually run the script to see if it behaves as expected.



Writing scripts (CCL) with MicroPhone is easy as 1, 2, 3, 4...

Figure 5.17 MicroPhone Application Generator



Writing scripts with MicroPhone's 'Watch Me' is even easier—its automatic!

Figure 5.18 MicroPhone 'Watch Me'

In programming, a variable can be thought of as a temporary storage box in memory that can be given a label. When you want to store something in the box, you do so by referring to its label. Likewise, when you want to retrieve the contents of the box, you use the name, that you give the variable when you first set it up. *String variables* store ASCII characters; *numeric variables* store numbers.

Variables are useful when you want to save a response from the remote system that will change from session to session, but one that will be needed more than once during a session. A variable may also be used to store a temporary number for the duration of a script's operation. Both inTalk and MicroPhone let you define variables.

MicroPhone has a useful, built-in editor called the MicroEditor, which can be called up while you are on-line. You use the MicroEditor to compose messages rather than using the usually cumbersome, remote system's editor, if it has one. This is a really useful feature as anyone who has spent hours on bulletin board systems can attest. However, when a MicroPhone script is executing, you cannot access the MicroEditor, even if you manually put the script on pause. This is a slight drawback. It means that you can't create a script that will stop at some point in a session and let you compose a message. Rather, you must create two separate scripts: one for execution before the point at which you want to use the MicroEditor, and one to take over after you've composed your message. Of course, you must manually start the second script when you've finished editing.



On-line Debugging

Any time you are writing a program meant to be performed in tandem with responses from a remote system, you have to test the program while connected with that system. If the program is extensive and complex, this can take a great deal of on-line testing time and should be factored into any cost estimate you might make for developing a program. For example, preparing our relatively simple example scripts required about 50 separate phone calls to the test Tree. That's another reason we stuck with a simple example: it costs less to dial into a system that does not charge connect-time fees.

MicroPhone and inTalk both come with predeveloped scripts that will let you log on to the more popular services, such as CompuServe and the Source. You can look at these scripts to see how they were done, then use what you learn in developing your own custom programs.

inTalk

The most complete set of control structures built into a terminal package belongs to inTalk. It is therefore the most comprehensive and flexible of the ones we have seen to date. However, that makes it correspondingly more difficult to master. It does not have a program recording or application generator function. Experienced programmers will find its "write, run/debug, edit, run. . ." cycle to be a familiar one. As shown in Figure 5.19, debugging is made easier by the fact that, as lines of code are executing, inTalk displays the current line at the bottom of the screen as you watch the on-line dialog in the main window. Programs can be written or edited with a word processor.

Novice users who want to move into the power ranks will find inTalk commands easy to understand. Fully exploiting them, however, requires substantial time and energy expenditures for all but the most experienced telecom programmers.

With inTalk you can't *nest* your programs, which means that you can't have your program call up another program for execution and then return to the original calling program. Instead, inTalk uses sub-routine calls within a program to accomplish the same thing. It also

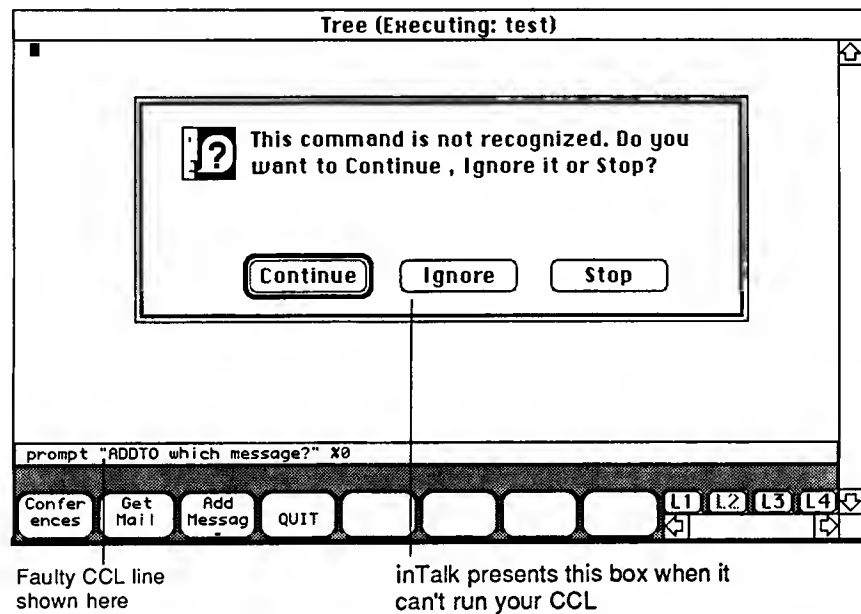


Figure 5.19 inTalk Debugger

enables you to set up a program file, which is similar to a batch file. The program file contains a list of programs to run one after another. This is similar to calling a program from inside a program, but you can't return to a previous program without specifically executing it again within the program file.



Host Mode

If you want to use any of these programs to turn your Macintosh into a dial-in host computer, there are several factors to keep in mind.

First, you may not want just anyone dialing in. This means that the host mode should be password protected. MicroPhone uses a single-password scheme, which is adequate for small, tightly-knit groups, provided the group password is changed frequently. Smartcom and Red Ryder also use a single-password lockout.

In its host mode, inTalk will only connect, send a file, or receive a file. It will operate unattended in its host mode. However, you cannot ask the calling user for a password, or otherwise conduct a dialog with him or her. Furthermore, your caller must know the name of any file he or she wants inTalk to send; it will not display a menu of files, nor will it allow you to lock out individual files. Once the caller has access to your system, he or she has full access. On the other hand, the caller will not be able to remove or alter existing files. None of the foregoing will make much difference, actually, unless you want to run a full-fledged BBS in your basement and let anyone access it.

Red Ryder

Note: Why Red Ryder? Scott Watson's explains. "Many of you have asked me why in the world it's named Red Ryder. It's the funniest story in the world and the only secret I won't tell, as a promise to Wat Buchanon. [A friend of Watson's who died. -Ed.] But, for you super-hackers, the answer is buried quite cleverly in the software. He would have wanted it that way. No hints, no way." This gives you the full flavor of Red Ryder's "personality."

Red Ryder is a shareware package for which the author asks payment from those who get and then actually use the program. Scott Watson, the programmer-entrepreneur who writes, distributes, and supports Red Ryder, does more than ask for payment in the polemics sprinkled through Red Ryder's documentation. Watson is a missionary, that is, a man with a mission, and it's nothing less than recruiting you into the growing ranks of those who are fed up with commercialism in software. For \$40, he'll give you the "inside poop" . . . and more.

Red Ryder's Write A Procedure For Me (Figure 5.20) mode is similar to MicroPhone's Watch Me. In fact, the only way to create a procedure (program) within Red Ryder is to use this feature. It has no built-in editor. You can, however, write procedures with your own word processor which can then be changed to Red Ryder procedures.

This lack puts an extra step into the normal "write, run/debug, edit, run. . ." cycle. Before you can edit a Red Ryder procedure, you have to change it from a procedure to a source file. Before you can run it and debug it, you have to change it back again into a Red Ryder procedure (or, to use Watson's term, remote service procedure). For lengthy and/or complicated programs, this gets tedious very fast.

The language lets you create macros easily. It also lets you create a host system that can take calls, dialog with users, and otherwise allow callers access to your Macintosh disks.

Finally, Red Ryder does not have quite as complete a set of execution control structures as inTalk or MicroPhone has. It does allow branching and nested procedures (the ability to start up a new program from inside another) but no logical condition testing, such as IF THEN ELSE branching.

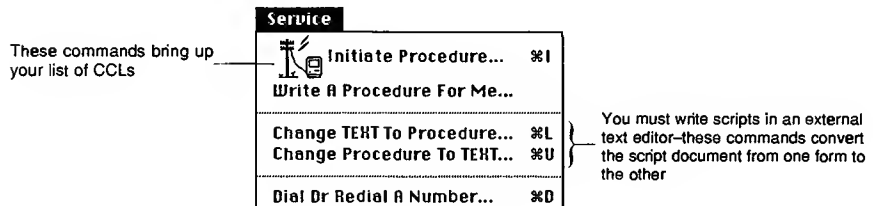


Figure 5.20 Red Ryder 9.4 Menu

Smartcom II

Smartcom II from Hayes is good intermediate package. Its Autopilot language is barely more than a macro command set. We could not write a program equivalent to our sample programs, here, in our tests. Perhaps offsetting its lack of features in its command language, Smartcom II uses the Macintosh interface better than most terminal programs we've seen.

As shown in Figure 5.21, Smartcom II has the ability to transmit almost-real time, quasi-interactive graphics with someone else who's also running Smartcom II on their Macintosh. This additional visual feature is something none of the other packages we looked at has. This ability may well offset its relative lack of language features, depending on your interests.

Comparing Languages

The list of features shown in Figure 5.22 provides an overview of the two languages that were capable of handling our relatively simple programming example. Because neither Red Ryder nor Smartcom provided us with a means of creating dialog boxes for local input or a way to prompt the remote user for conditional options, we couldn't create our test program with these packages.



Qué CR, CR

A relatively minor but notable difference between inTalk and MicroPhone has to do with the way they treat lines typed into their scripts or sent to the remote system. InTalk automatically appends a carriage return (CR) at the end of its lines, so you needn't specify them when you are writing a script. You can tell inTalk *not* to do this if you want, but the default is that it will. With MicroPhone, however, every CR you want sent to the remote system must be called out in your script.

Summary

Because of innumerable possibilities, designers cannot anticipate every single end-use case, and users must do the final work of "educating" their programs. There are now several good-to-excellent alternative, general communication terminal programs for the Macintosh on the

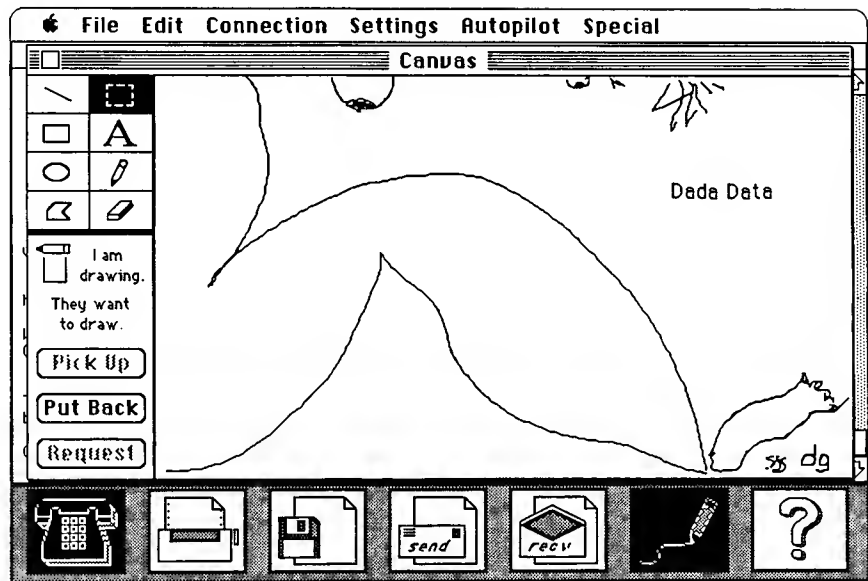


Figure 5.21 Smartcom II's Graphic Box

This is our comparison chart of scripts: MicroPhone v. inTalk:

<u>inTalk</u>	<u>MicroPhone</u>
<u>Telephone/Modem Control</u>	
Dial	Dial service ...
Retry n times	
Delay n seconds	Wait for call
Hangup	Hangup
<u>Program Flow Control</u>	
Load <file>	
Wait ... <event> Wait for text	
Delay	Seconds
Echo <char>	
Resume	
Prompt	
Quiet	
String	For line containing ...
Until time	Until time

Figure 5.22 Side-by-Side Comparison

inTalk

If ... <not> ...

When
StringQuiet
#Disconnect
When CancelCollect Until \$n
Concat \$n
Decrement/Increment
Set <variables>
10 each numeric &
string variables
Functions keys -- 32
Execute <file>
Goto <label>
Perform/Return
Cancel
Quit
Restart**MicroPhone**If ...
Text equals
Line contains
Line doesn't contain
Success
Failure
Else If ...
EndifWhen
Text equals
Line contains
Line doesn't contain
Success
Failure

End When

While ...
Text equals
Line contains
Line doesn't contain
Success
Failure
End WhileRepeat
Until
Text equals
Line contains
Line doesn't contain
Success
FailureCycle (restarts While or
Repeat loop)Do Script
Chain to Script
Return**Figure 5.22** (continued)

<u>inTalk</u>	<u>MicroPhone</u>
<u>File Transfer & Control</u>	
File Rcv Text	Open Capture File On/Off
File View Text	
File Close	
File Pause	
File Resume	
File Send Binary	Send File
	Macbinary
	Xmodem
File Send Text	Text
	Line-by-Line
	Macterminal
File Rcv Binary	Rcv File <Xmodem only>
Send <string>	Send Line
	Skip Line
<u>Printer Control</u>	
Print On/Off	Printer On/Off
<u>User Interaction</u>	
Clear	
	When Button A/B/C branching
Display	
	List in Menu
Level <Function Kay Macros>	List in Button
	Send Text ... <from dialog box>
Prompt	
Keyboard Lock/Unlock	
<u>Remote System Interaction</u>	
Send <string>	Send Text String
	Send Protected String
<u>Programming Support</u>	
No Show	Print a Script
	Check Syntax
<u>Additional Features</u>	
Collect Until \$n	Protected Strings
Concat \$n	
Set <variables>	
<10 numeric and	
10 string variables>	
Function Keys (macros) <32>	
Assumes CR at end of all lines	CRs must be explicit
Can "nudge" script from keyboard	Keyboard always locked while
while program is executing	script is executing

Figure 5.22 (continued)

market that let users customize their operations through a combination of macros and a Communications Command Language. Each has a slightly different combination of elements and each shows strengths for particular classes of user.

For those who use telecom infrequently, either MicroPhone or Smartcom II will do nicely. Both have all the features plus high “friendliness quotients” which will be appreciated most by casual users and beginners. In corporate settings where training time is at a premium, these ease-of-use packages will also be appreciated. Smartcom II has the added attraction of its interactive, MacPaint-like on-line graphics exchange.

For business users who use telecom *a lot* in their work, inTalk or MicroPhone are the packages of choice. MicroPhone scripts are easy to create and modify. Its Watch Me feature literally seduces you into its scripting language. Both inTalk and MicroPhone support very high-speed data transmission with Telebit’s Trailblazer modem. This means that high-volume information transfers can be made more economically.

Experienced programmers and those who customize communications packages for clients will find inTalk’s CCL to be flexible and powerful. Its Macintosh user interface may require a bit more training for naive users, but the assistance of a skilled programmer can smooth out its rough edges nicely.

Finally, if you want to be a part of a growing community of on-line comrades-in-telecom, and if you consider yourself to be a closet hacker, then Red Ryder is the program of choice. We found its vanilla terminal features to be quite useable, but we had the advantage of knowing the Hayes’ dialing command set and are accustomed to wading through less-than-transparent technical manuals.

Each of these programs has found a growing audience of users. Each embodies ideas about communication, about programming, and about the use of the Macintosh in slightly different ways. Whichever one you choose, you’ll find that the time you spend getting proficient with a telecom programming language will pay off more quickly than you may believe possible.

CHAPTER

6

Links and Hints

The winds are on the side of the ablest navigator.

Gibbon

In this chapter we provide information that will assist you when you're trying to link your Macintosh to just about anything else. We begin with a Macintosh-to-Macintosh link and work our way to main-frame connections.

In each section, we list terminal packages that we've found to work with the machines listed. There are lots of possible hardware/software combinations we couldn't possibly cover. However, this chapter and its guideline examples should enable you to connect with any micro, not just the ones we specifically mention. The comparison chart at the end of the chapter will help you sort out terminal package features.



Macintosh to Macintosh

Cables

All cable figures refer to the cable diagrams in Chapter Eight.

Direct connect cable: Figure 8.8e

Smartmodem cable: Figure 8.8a

Apple modem cable: Figure 8.8b

Apple Personal Modem cable: Figure 8.8c

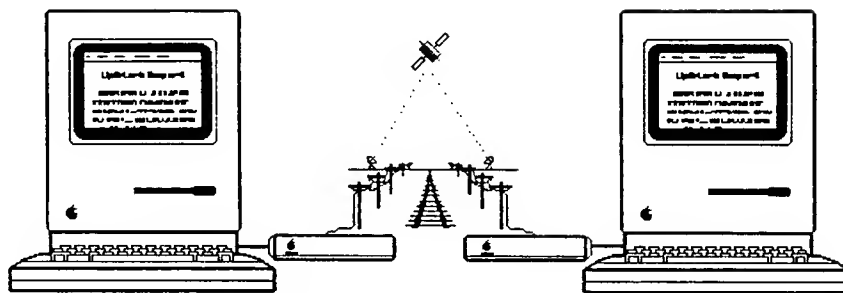


Figure 6.1 Macintosh to Macintosh

General Discussion

Either or both machines may be Macintosh Pluses. If so, you must make sure the proper cables are in place. Macintosh Plus serial ports use their own, idiosyncratic, eight-pin circular connector instead of the normal DB-9 connectors used up until now by the Macintosh. Connecting to the Macintosh Plus ports requires a special cable or the use of adaptor cables that go between your existing cables and the Macintosh Plus. Chapter Eight has more on this subject.

They may be connected directly using one of the cables specified. You may want to do this because you have about a megabyte of document files on your hard disk drive that you want to send to someone else's Macintosh. By direct-connecting, you can transfer these files at 9600 bps or faster with no time-consuming disk copying and swapping. With a permanent direct-connect, two Macintoshes in adjoining workspaces can exchange hard disk files.

The Macintosh may be connected through modems and the phone lines. This is the most common way to connect and the one we emphasize throughout *MacAccess*.

Finally, either or both of the Macintoshes must be running one of the terminal programs listed in the chart in this section, or some other terminal package of your liking. If both Macs are running the same program, all the better. Most of them have special selections that make terminal-to-terminal transfer especially easy if both machines are running identical packages.

If, on the other hand, the machines are running different packages, it won't make that much difference; nearly all the available Macintosh terminals support both regular Xmodem and MacBinary transfers. Only one package, BLAST, doesn't, but you probably won't

be using that anyway. We included it in this chart because, if both Macintoshes are running BLAST, you can get fast Macintosh-to-Macintosh, error-free, eight-bit transparent transfers using BLAST's transfer protocol instead of MacBinary or Xmodem. BLAST is best used for Macintosh-to-mainframe links. (See "Micro to Mainframe Software Solutions" in this chapter for more about BLAST.)

The modems listed in the chart are those specifically named as *supported* in the documentation that comes with the different packages. This does not mean that a modem not mentioned will not work with the package. It just means that you'll have to experiment to discover what works and what doesn't. Usually, a call to either the terminal software developer's customer support line or to the modem manufacturer's customer support service will get you the answers you need to know about software/modem compatibility. We should also mention that we have tested only a small fraction of the hundreds of modem and software combinations available to us. In order to have tested *every* combination and permutation, we would have required the resources of the Federal budget.

The Macintosh A column contains the name of the package, and the modems and file-exchange protocols supported. The Macintosh B column repeats the name of the package, but it does not repeat the modem or protocol information, which are exactly the same as those for Macintosh A. To link any two Macintoshes, then, determine which terminal package is available to run at either end and the protocols associated with that package. Since most of them support MacBinary exchanges, that is the obvious choice in most cases.

Even when trying to link with another brand of machine, we suggest checking the chart first to find a compatible protocol. Then check the Macintosh column in the later, machine-specific charts. In those sections, we show only those additional Macintosh packages that are designed to work with that particular other machine.

Whenever you want to link with a new machine, follow these steps:

- First, determine which protocols are supported at the other end of the link. For example, let's say you find out that the other machine will handle Xmodem but not MacBinary. (You find this out by asking the owner or operator of the other machine to tell you.)
- Next, determine which Macintosh terminal package will support the protocol used by the other machine. Since Xmodem is quite popular, nearly every package in this section will support it.

- If you already have a package that supports the protocol you want to use, you're all set. But if there are no compatible protocols available, then one or both of the parties who want to exchange files will have to first change their communications software. As far as the other party is concerned, we have included in our charts for other machines only the most popular and/or the most Macintosh-friendly terminal software packages.
- Once you decide to buy or change your terminal software package, you'll want to make sure that it works with your modem before you purchase it. If you're buying both for the first time, try to base your decision about which modem to use on the software you want. (In other words, don't buy a modem just because "free" terminal software comes with it.)

Note, further, that all the Macintosh packages listed in the chart are capable of ordinary vanilla ASCII text transfers. For a more complete comparison of the popular Macintosh packages, see the chart at the end of this chapter.

Macintosh-to-Macintosh Chart

Note: The numbers in the chart refer to the notes that follow the chart.

Mac A		Mac B
MicroPhone		MicroPhone
MODEMS:	Apple Hayes & compatibles ¹ Intelligent ² Telebit ³	
PROTOCOLS:	MacBinary MacTerminal Xmodem ⁴ Ymodem Telebit	
inTalk		inTalk
MODEMS:	Apple Hayes & compatibles ² Datec Multitek Prometheus U.S. Robotics Ventel Intelligent ² Telebit/DCA Fastlink ³	
PROTOCOLS:	MacBinary MacTerminal Xmodem	

Mac A		Mac B
	Crosstalk ⁵ inTalk ⁶ Telebit RLE/Vidtex ¹¹	
Smartcom II		Smartcom II
MODEMS:	Apple Hayes & compatibles ¹ Intelligent ²	
PROTOCOLS:	MacBinary MacTerminal Xmodem Hayes (proprietary) ⁸ Graphics ⁹	
Straight Talk/Dow Jones		Straight Talk/Dow Jones
MODEMS:	Apple Hayes & compatibles ¹ Intelligent ² Novation 103/212 Smart Cat POPCOM X100	
PROTOCOL:	ASCII Text only	
Red Ryder		Red Ryder
MODEMS:	Apple Hayes & compatibles ¹ Intelligent ²	
PROTOCOLS:	MacBinary Xmodem ¹⁰ Kermit RLE/Vidtex ¹¹ CompuServe 'B' ¹²	
PC to Mac and Back		PC to Mac and Back
MODEMS:	Apple Hayes & compatibles ¹ Intelligent ²	
PROTOCOLS:	Xmodem PC-Mac and Back (proprietary) ¹³	
MacTerm (Sidekick)¹⁴		MacTerm
MODEMS:	Apple Hayes & compatibles ¹ Intelligent ²	
PROTOCOLS:	ASCII Text only	

Mac A	Mac B
Telescape	Telescape
MODEMS:	Apple Hayes & compatibles ¹ Intelligent ² Novation Smart Cat Plus
PROTOCOLS:	MacBinary Xmodem (variable timeout & CRC)
Terminal (Jazz)¹⁵	Terminal
MODEMS:	Apple Hayes & compatibles ¹ Intelligent ²
PROTOCOLS:	Xmodem Jazz ¹⁶
MacTerminal 2.0	MacTerminal 2.0
MODEMS:	Apple Hayes & compatibles ¹ U.S. Robotics Autolink 212A Intelligent ²
PROTOCOLS:	MacBinary MacTerminal Xmodem ¹⁷
VersaTerm	VersaTerm
MODEMS:	Apple Hayes & compatibles ¹ Intelligent ²
PROTOCOLS:	MacBinary MacTerminal Xmodem Kermit
MacMail/MacGeorge	MacMail/MacGeorge
MODEMS:	Apple Hayes & compatibles ¹ U.S. Robotics Racal-Vadic's "Maxwell Modem" Intelligent ²
PROTOCOLS:	MacBinary (includes CRC) Xmodem ¹⁰

1. See Chapter Three, "Modems."
2. An *intelligent* modem is one that can perform operations under control of signals from the computer. (Hayes-compatible modems are all intelligent in this sense.) You must know the command set of your modem and issue these commands either directly from the keyboard or by preprogramming macros, with the appropriate modem dialing and connect commands.
3. See Chapter Eight.
4. Supports 1K block transfers and Cyclic Redundancy Checking (CRC).

5. See Chapter Eight, also Macintosh to IBM PC in this chapter.
6. Palantir's proprietary internal protocol used for error-free exchanges of complete Macintosh documents (both Data and Resource Forks). It allows file transfers to be controlled by one operator provided the remote system is also using inTalk protocol. You must know the file name to receive a file using this protocol.
7. The automatic dial functions work with the Hayes modem *only*. To use Hayes compatibles, you must specify Direct Connect and then manually address the modem using command codes. See Chapter Three, "Modems."
8. Hayes' proprietary protocol.
9. Smartcom II also has an interactive graphics mode, whereby two people can exchange graphic information by drawing on a limited MacPaint-like screen. Only one person can draw at a time, and Smartcom II *must* be running on both Macintoshes. This feature may appeal to people who must work on tight deadlines with graphic-oriented material.
10. Supports Cyclic Redundancy Check (CRC).
11. A graphics protocol used by CompuServe.
12. Another CompuServe protocol.
13. Special proprietary protocol to transfer files between the Macintosh and the IBM PC. (See "Macintosh to IBM PC.")
14. This little desk-accessory is suitable for sending quick text-only memos to various electronic-mail destinations. It would be more useful if it supported Xmodem and MacBinary! See Chapter Seven for further discussion.
15. This terminal program is part of Jazz's integrated software package.
16. Lotus' proprietary protocol for error-free exchanges of complete Mac documents (both Data and Resource Forks). The other Macintosh must also be using Jazz.
17. Includes Text Xmodem.

Macintosh to Macintosh Through an Intermediate System

In some cases, you may want to use an intermediate system to send Macintosh information to a second remote Macintosh. Maybe you want to send the same Macintosh application you've written to a whole group of others. For convenience, you decide to use an intermediary

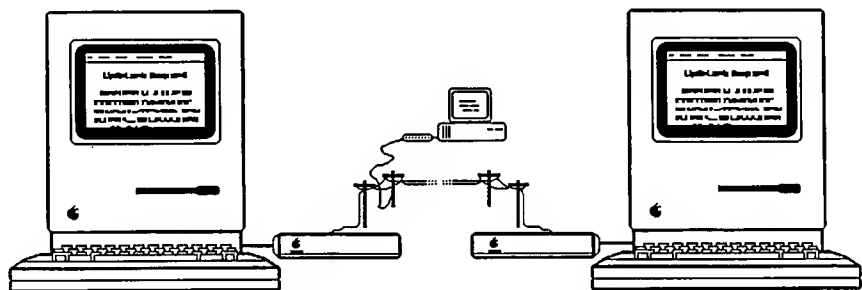


Figure 6.2 Mac to Mac via Intermediate

system that is always on line and waiting to serve dial-in users. That means using your modem and phone lines to send a Macintosh file to the remote intermediary system. At their convenience, the other Macintosh users dial into the intermediate system and capture what you've sent. (Direct cable connections to intermediate systems are covered later in this chapter.)

How you deal with the intermediate system depends on whether or not it supports either of the two Macintosh transfer protocols: MacBinary or Macterminal. If it does, you're in business as long as you have a Macintosh terminal program that supports one or the other. If the remote system doesn't support either of these, then you'll have to ask more questions.

Perhaps the remote intermediate supports Kermit, which is available now on many mainframes, especially at universities. If so, and if your terminal package doesn't support Kermit, then you will have to acquire one of the packages listed in the chart that also supports Kermit (VersaTerm, for instance).

If the remote intermediate will accept only ASCII text, with no error-checking, it is still possible to exchange Macintosh files with another Macintosh. It just requires two extra steps, one at each end of the transfer.

For the sake of example, let's assume that two Macintosh owners want to swap MacPaint documents through a host system running on the IBM PC. The PC host system will accept ASCII text files *only* (no MacBinary or MacTerminal transfers) and store them as regular IBM PC DOS text files.

Here's the general procedure for file exchange between Macintoshes using a system that does not support much other than ASCII exchanges:

1. Convert your MacPaint or other Macintosh document into a consolidated (both data and resource forks) ASCII file with BinHex 5.0. (See Chapter Eight for information about this program.)
2. Log on to the remote intermediate system and send the converted Macintosh document using plain ASCII text transfer.
3. Your associate "grabs" the file from the remote PC or mainframe, again using a text transfer.
4. Said associate then must *reconvert* the file using BinHex 5.0, so that it is once again a legitimate Macintosh document. The document is restored and can then be opened by a Macintosh application. Macintosh applications, themselves, may also be transferred this way.

and can then be opened by a Macintosh application. Macintosh applications, themselves, may also be transferred this way.

This BinHexing trick works well with just about any intermediate system, but it has the disadvantage of that extra step at both ends of the transaction to convert and reconvert the Mac file. Incidentally, if the remote system supports Xmodem transfers, then the BinHexing approach will still work. Just use Xmodem instead of ASCII transfer.

If you want to use your office PC to exchange Mac documents, but you don't want to hassle with BinHex, there is yet another alternative. Set up the office PC with MacLink attached to a Hayes or Hayes-compatible modem. *Note that MacLink must also be running on the Macintosh.* This combination is a powerful one. Anyone dialing into the PC using MacLink gets control of the PC's DOS file functions. (It won't let you erase files, of course.) You can change logged disks, send files, and receive files. Moreover, with MacLink, you don't have to worry about file conversions in either direction. You can exchange Macintosh documents with other Macintoshes, send Macintosh word processor documents to the PC for use with PC word processors, and send PC word processor documents to the Macintosh. All the relevant formatting information is automatically sent!

This combination serves both Macintosh users and IBM PC users; however, it is a closed system. You don't get to dial or communicate with anything other than another system running MacLink.

PC to Mac and Back, shown in the Macintosh-to-IBM PC chart, doesn't have this limitation. You can dial any other system using PC to Mac and Back. However, this product does not have as many file-conversion options built into it. See "Macintosh to IBM PCs" later in this chapter.

When using a mainframe as an exchange point between two Macintoshes, transfer options will depend on what the mainframe supports. Many, as we said earlier, support Kermit, which means Macintosh-to-Macintosh exchanges are easy, provided that both Macintoshes use a package which supports Kermit.

In the UNIX world, two programs are widely available that can be run remotely in order to exchange Macintosh files in a MacBinary format. Look for *macput* and *macget* on your remote UNIX system. If these are available, you're in business; otherwise, use BinHex.

There may also be some Macintosh utilities on your VAX/VMS remote system. The common name for it is XMAC. Again, if you are familiar with this environment, all you do is run XMAC on the VAX and exchange your files using MacBinary on the Macintosh.

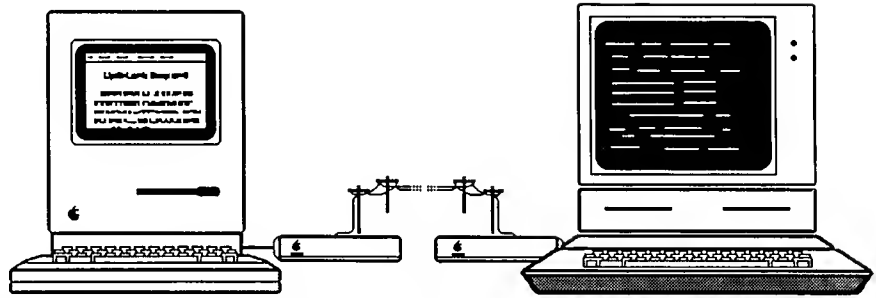


Figure 6.3 Macintosh to Apple II

Macintosh to Apple II Family

Cables

All cable figures refer to the cable diagrams in Chapter Eight.

ImageWriter cable: Figure 8.8d

Smartmodem cable: Figure 8.8a

For this discussion, Apple II means any member of the Apple II family: Apple II, II+, IIe, IIfx, or anything else that is likely to come along in the next few years. The direct-connect cable assumes the use of any of the Asynchronous Communication Cards available for the Apple II. These come in two “flavors”: one uses a regular RS-232 modem cable, the other type needs a null-modem cable. You must check the manual for your particular serial card to find out which you need. The difference lies in whether the card behaves as a DTE or DCE (see Chapter Four). How you connect your Apple II to your modem depends on which modem you are using and other internal Apple II factors which we can’t possibly cover here.

Suffice it to say that we can vouch for the fact that each of the packages listed in the previous chart can exchange text files with the Apple II family running the appropriate terminal program.



Macintosh-to-Apple II Chart

Note: The numbers in the chart refer to the list that follows the chart.

Macintosh	Apple IIe
[see Macintosh-to-Macintosh chart]	ASCII Express Pro ¹
	Xmodem
	Data Capture ¹
Mac.Transfer ²	Mac.Transfer
II in a Mac ³	II in a Mac
Mac + II ³	Mac + II

1. Plain ASCII exchanges via modem. Data Fork text only.
2. Direct-connect using null-modem cable or use modems. Mac.Transfer is not a general-purpose modem program. Rather, it performs the file conversions necessary to share text between Apple II word processors and Word or MacWrite. SYLK and DIF files are also massaged and arrive at the Macintosh (or back at the Apple) ready for loading into Excel or other spreadsheet programs. Mac.Transfer for the Apple II family is similar to MacLink on the IBM PC family.
3. Both II in a Mac and Mac + II allow you to make your Macintosh emulate an Apple II. They make your Macintosh capable of running many existing Apple II programs. Both come with complete instructions for transferring application programs from the Apple to the Macintosh, including cable wiring and hookup directions. These are *not* general terminal programs. Again, they are designed to work with copies running on both sides of the transfer. Each package comes with a setup disk for the Apple II and a disk for the Macintosh.

Macintosh to IBM PCs

There are as many terminal programs for the IBM PC as there are for the Macintosh—perhaps more. Crosstalk is one of the most versatile and most widespread of the commercial packages. PC-Talk and Q-Modem are popular shareware products. Your local IBM PC users' group probably has many additional telecom shareware programs available. Most of the popular PC terminal programs support plain Xmodem at least (a version that doesn't employ CRC or provide for variable "relaxed" timing—see Chapter Eight).

Cables

Cable figures refer to cable diagrams in Chapter Eight.

MacLink cable: Figure 8.8f

PC to Mac and Back cable: Figure 8.8g

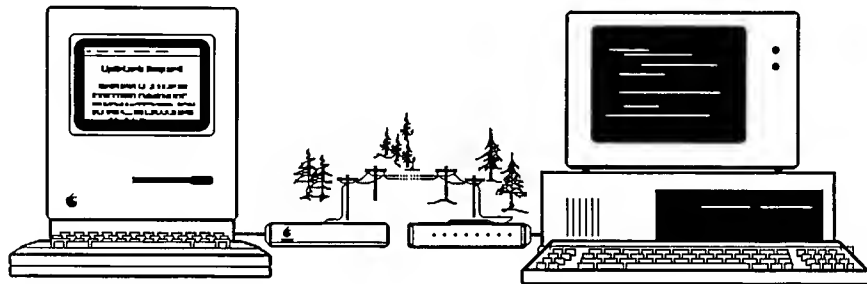


Figure 6.4 Macintosh to IBM PC

Macintosh-to-IBM PC Chart**Macintosh****IBM PC**

[see Macintosh-to-Macintosh Chart]

PC-Talk III¹Q-Modem¹

inTalk

Crosstalk or inTalk

MacLink

MacLink

PC to Mac and Back

PC to Mac and Back

1. These are general purpose terminal programs running on the PC. They can be used to exchange a Macintosh text file as is, or, using BinHex 5, they can be used to transfer applications and documents from Macintosh to Mactinosh.

**The Wonders of MacLink**

MacLink (DataViz) is specifically designed to handle the sticky stuff of file translation between the Macintosh and the PC. It must be used on both machines. Additionally, MacLink cannot be used as a general purpose terminal program to dial other systems or services. It is strictly a “run-with-copies-of-itself” program.

Just about any word processing program your PC colleague is using can be translated into a form useable by the Macintosh. With MacLink at both ends of the transfer, the following conversions are handled as the transfer takes place:

Lotus 1-2-3 to Multiplan

Lotus 1-2-3 to DIF

Multiplan to Lotus 1-2-3

Multiplan to Multiplan

Multiplan to DIF

DIF to DIF

DIF to Lotus 1-2-3

WordStar to MacWrite

MacWrite to WordStar

MultiMate to MacWrite

MacWrite to MultiMate

MacWrite to Text

DCA (DisplayWrite 3) to MacWrite

Text to MacWrite

1-2-3 to Excel or Jazz

Word to Word

The most notable omission from the above list is that there is no direct translation between Word (on the Macintosh) and WordStar (on the IBM) files. You can get a WordStar file into Word by using MacLink's WordStar to MacWrite conversion and then opening the MacWrite file with Word on the Macintosh (whew!). But you can't go the other way around: MacWrite won't open Word files, so you can't use the MacWrite to WordStar conversion to get a Word file from the Macintosh into WordStar on the IBM. (Word 3.0 for the Mac should fix this problem—it will both read and write MacWrite files.)

MacLink can be used either through Hayes or Hayes-compatible modems with the phone lines, or, better yet, through a direct-connect cable. We recommend the latter if the two machines can be conveniently moved into proximity, especially if you want to transfer long files. Direct connection lets you transfer data at a much higher rate (9600 bps instead of 1200).

MacLink is especially useful for sharing spreadsheet data. For example, let's say you are trying to send a Lotus 1-2-3 file to the Macintosh for use with Multiplan. As it converts the data, MacLink will generate the equivalent formulas for Multiplan using the 1-2-3 data, and insert them into the proper cells in the bargain. Formatting information is also passed along, so that the target table will look as much like the source table as possible.

Finally, even if the translation you want to make isn't directly possible with MacLink, it may still be possible to do, depending on what software package you're using on the IBM PC. For example, the relational data base program, Paradox, is able to import and export a variety of file types, including Lotus 1-2-3 data. If you want to exchange information between Excel on the Macintosh and Paradox, for example, all you need to do is export Paradox's table to a 1-2-3 *.wks* or *.wk1* file, then use MacLink to move it to the Macintosh. Similarly, if you want to move a file into Paradox from the Macintosh, use MacLink to move the file to the PC, then import the file using Paradox's own built-in translators. Many application programs now allow this kind of translation. For more on this topic, see our file import/export information in Appendix A.

Additional Comments

If you need a general-purpose terminal program as well as file-translation capability, you might want to use PC to Mac and Back. However, its overall features are not very sophisticated, and it cannot handle as many different kinds of source and target files as MacLink can. It can

do the basic WordStar-to-MacWrite kind of conversion, but not “on the fly.” You have to run PC to Mac’s file conversion utility before you attempt to exchange files. (This is similar to having to use BinHex at both ends.) If your needs are confined to word processor text, this might be the program for you. Our recommendation for greatest flexibility, however, is to use a combination of a powerful general-purpose program from the Macintosh-to-Macintosh chart with the equally powerful file transfer capability of MacLink. Use a general-purpose package that supports MacBinary or MacTerminal protocols for your Macintosh to Macintosh transfers, and use MacLink for Macintosh to PC transfers.

If you are not using MacLink, you may need to have the PC side massage its files before sending them, using yet another file conversion utility: Software Bridge. Software Bridge will import and export files for the following popular IBM PC family of word processors:

MultiMate

DisplayWrite II and III

Samna Word

WordStar

Word Perfect

Word Marc

DCA (IBM’s Document Content Architecture)

ASCII

The codes included in the translations done by Software Bridge provide just about everything you’d need to preserve hard and soft carriage returns, tabs, underlines, boldface, superscripts and subscripts, margins, spacing, and indents. This utility only runs on the PC family, but it can be used to convert before shipping the file elsewhere. After all, the translation program doesn’t care what your *real* target machine is anyway.

WordStar

WordStar is one of the most widely used of all microcomputer word processors, so the amount of information stored in WordStar files is enormous. Again, if you use MacLink over either a modem or direct-connect Macintosh-to-PC cable, there’s no problem. MacLink will take care of—and preserve—all of WordStar’s formatting codes.

However, let's say you just tried to send a text file to a PC without using a file-translation program. Someone at the PC side is going to try to edit the file with WordStar. Let's say you just sent a file which you first saved as a text file through Word.

Now, your associate looks at the file with WordStar and sees what looks like minus signs or hyphens at the 80th column of each line of text on the screen. What's going on here?

WordStar uses the hyphens to show the place where there are only carriage returns. Normally in WordStar files, the symbol < appears at the end of each paragraph. This is WordStar's way of representing a carriage return/linefeed combination (CR/LF, ASCII 13 and 10). If WordStar shows a J in the far right column, it stands for ^J. You can think of it as "Just" a linefeed character (LF, ASCII 10).

If you attempt to print a WordStar file that only has carriage returns in it, the printer will overprint each line. The printer needs a linefeed to move the paper to the next line, and it needs a carriage return to begin printing back at the left side of the paper. Therefore, either the file has to be fixed by hand before it can be printed, or steps have to be taken to make sure that the lines being sent end with a CR/LF combination. This is where the transmitting program's smarts come in. If your terminal program allows it, you can transmit the text and specify that the program put in the right CR and/or LF combination at the end of each outgoing line.

To fix the WordStar file by hand at the IBM PC side, get into WordStar and perform the following steps. In this list, the sequence shown as ^A, for example, means "control A," a simultaneous key-press of the Ctrl key and a letter key.

1. Press ^Qa to tell WordStar to search and replace.
2. When prompted for the search string by WordStar, press ^P ^M <Return>. The ^P tells WordStar that the next character is an embedded control character to search for. The ^M, of course, is the return character.
- 3a. When prompted for the replacement string by WordStar, enter ^N <Return>, which is "WordStarese" for the CR/LF pair.
- 3b. OR, you can replace all the CRs with a plain space, making it easier to automatically reformat the paragraphs. To do so, when WordStar prompts for the replacement string, enter a space <Return>. The ends of paragraphs, of course, will have to have the CR/LF pair.

There are several public domain freeware conversion utilities for going from ASCII to WordStar and back again. Check your local IBM PC users' groups.



MacCharlie

MacCharlie (Dayna Communications) is one of those special cases that doesn't fit easily into any category. It is a hardware solution to the problem of connecting a Macintosh to an MS DOS (IBM PC compatible) machine so that they can exchange information. It is a sophisticated and correspondingly costly solution to file sharing between the Macintosh and the IBM PC worlds.

The idea seems simple enough at first glance: link the PC-compatible world with the Macintosh world through a piece of dedicated hardware that latches onto the Macintosh. If desktop real estate is at a premium, this solution only takes an extra 50% of space for the hardware (less room than, for example, the IBM PC itself).



Photo courtesy of Dayna Communications

Figure 6.5 MacCharlie

The Macintosh display screen and keyboard are used as the I/O (input/output) devices for MacCharlie. A kind of “keyboard collar” is put around the Macintosh keyboard to make the layout and available keys conform to the familiar PC configuration. This expands the Macintosh keyboard footprint and may make mousing a bit more awkward.

The MacCharlie hardware consists of an 8088 microprocessor with one or two double-sided 360K disk drives and 256K to 640K of RAM. There are two RS-422 serial ports, one of which must be used to connect to the Macintosh’s printer port via a supplied cable. It also has one RS-232 serial port and an expansion chassis port in case you want to use any of the available PC add-on cards on the market.

In practice, the PC DOS prompt and interaction is “contained” in a Macintosh window. The menu bar and functions are shown in Figure 6.6.

The main advantage to this setup is that you’re able to run DOS programs and Macintosh programs side by side and share files between them. The suggested retail price of a 640K MacCharlie is \$995.

The MacCharlie approach is sometimes called *coprocessing*. The design takes care of all the internals for you and creates a smooth operating environment between the two. Menus, icons, software for linking and automatic file transfer are all contained in one seamless environment. Setup and operation are simple, and Dayna Communications’ customer support is superb, just in case you do run into a problem that you can’t overcome by yourself.

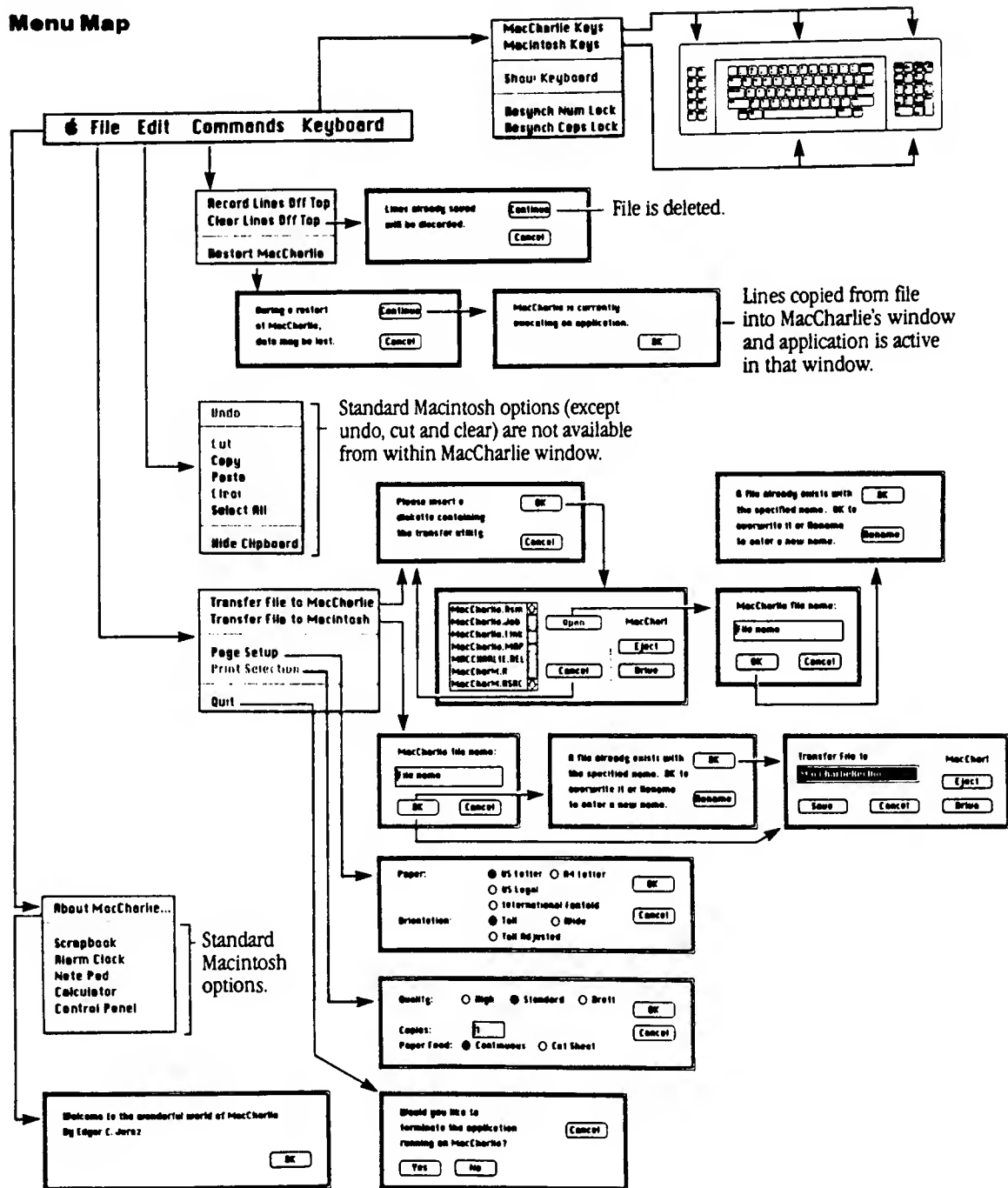
Another way of looking at this configuration is to consider the Macintosh to be an input/output terminal for MacCharlie.

What does MacCharlie let you do that the Macintosh or the PC alone won’t? For one thing, you can transfer sections or whole Word documents to WordStar, for example, through the Macintosh Clipboard. You can use it as a printer buffer for the Macintosh. You can use it to run programs that aren’t yet available for the Macintosh, such as Ansa Software’s Paradox. Finally, if you need a permanent solution to file-sharing and duplicate file creation between the Macintosh and the PCs in your office, MacCharlie is there, always hooked up and ready to go, with no need for additional special cables.

MacCharlie comes with its own set of file conversion programs, so you don’t have to worry about file formats. MacCharlie converts automatically upon transfer from the Macintosh to the PC or vice versa.

Aside from its speed in file transfers, some operations on MacCharlie are pretty slow, even for Macintosh users accustomed to some sluggishness from time to time.

Menu Map



Courtesy of Dayna Communications

Figure 6.6 MacCharlie Block Diagram

MacCharlie will run WordStar, 1-2-3, dBASE, Paradox, and Framework, to mention just a few of the more popular IBM PC programs.

If you are secretly in love with the Macintosh, even though you work for MegaCorp (or any other company that has “standardized” on IBM PCs or a compatible micro), then MacCharlie might be the ideal solution for you. Technically, what you will have is one computer that is PC-compatible that lets you use the Macintosh and still communicate with the PC world.



Abaton Drives

The Abaton 5.25 drive is a lower-cost solution to file sharing between the Macintosh and IBM disk file formats. In utility, it stands midway between a program such as MacLink, and a coprocessor such as MacCharlie. It includes a cable and necessary software. It doesn't let you run PC programs, though, the way MacCharlie does. It just lets you convert existing files on your IBM PC disks to the Macintosh format. If you have a CP/M computer, such as the Kaypro, Osborne, or Televideo, you can also use the Abaton drive to run software and convert files from these computers to your Macintosh. The suggested retail price of the Abaton at the time of this writing is \$695.

Macintosh-to-Mainframe Services

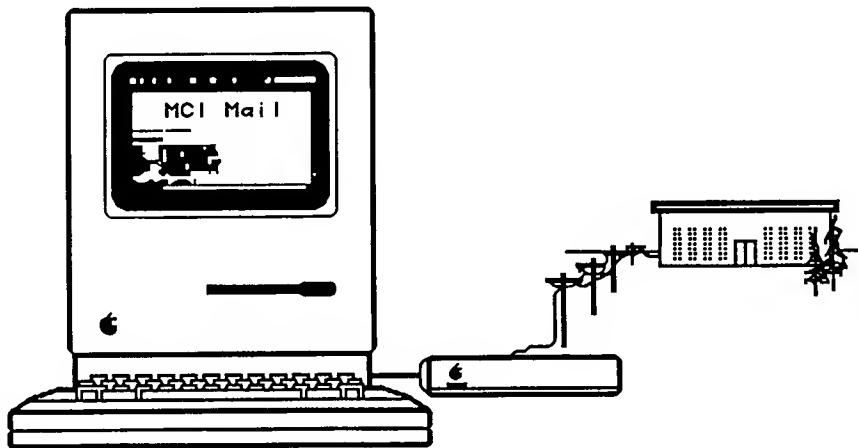


Figure 6.7 Macintosh-to-Mainframe Services

Any of the all-purpose terminal packages listed in the Macintosh-to-Macintosh chart will allow you to connect with the various mainframe information sources, such as Dow Jones News Retrieval, Newsnet, or CompuServe. CompuServe, for example, supports the exchange of files and applications between Macintoshes using MacBinary transfers and so does MCI Mail. You can customize general-purpose telecom programs to work with specific services using a built-in scripting language (as explained in Chapter Five) or through the use of macros. However, if you'd rather not have to do the customizing work yourself, there are several terminal packages that have been predesigned to streamline the use of specialized on-line utilities. Some, such as the Dow Jones Market Manager Plus, provide highly specialized features you won't find in any general purpose terminal program.

Dow Jones Market Manager Plus

This package is designed to work with Dow Jones News Retrieval to automate stock portfolio management. It dials into the Dow Jones data base of stocks and gets the prices on any of more than 6,000 securities listed. It can also get news stories from *The Wall Street Journal* and *Barrons*. The communications side of this package is limited to use with the Dow Jones service. That is not its main attraction; rather, it provides a sophisticated stock management system that includes on-line retrieval of information and automated stock portfolio management.

Market Manager Plus lets you maintain your cash balances for your portfolios (up to 26 separate portfolios), keeps a record of your transactions, and provides an audit trail. It also provides reports for your holdings by portfolio or security, and reports gains or losses from completed transactions. Reports are also provided for interest, dividends, and cash balances. The package works with Apple and Hayes modems.



The Meter Is Running

Time that you spend on line with any commercial service, such as the Source or CompuServe, is metered. That's why it's important to have your message ready to go before you log on. (MCI is an exception, but even with MCI you gain more efficient use by composing off line.)

Electronically generated mail that results in the delivery of printed copy is cheaper than sending mail overnight through services such as Federal Express or Express Mail, yet faster than just dropping a letter in a mailbox.

Macintosh to Mainframes and Minis

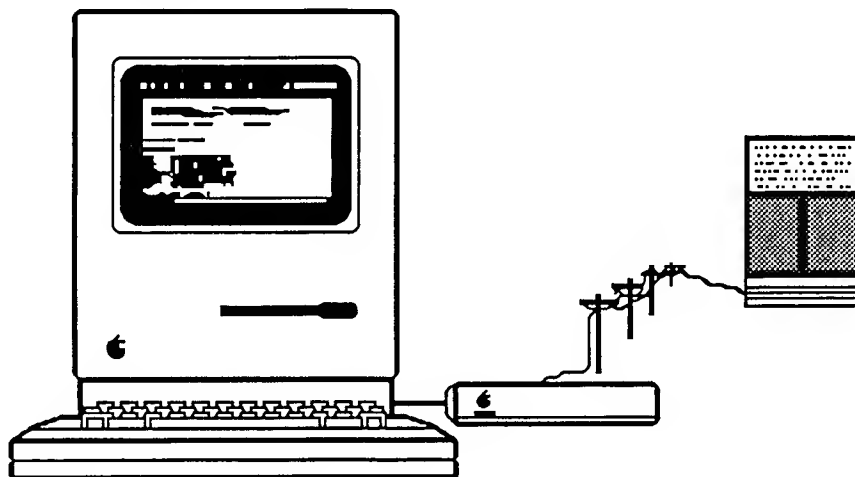


Figure 6.8 Macintosh to Mainframe

A mainframe system is any large computer (bigger than you can pick up and throw) owned and operated by a major institution (corporation, university, government), costing more than \$20,000. A minicomputer is midway between a micro and a mainframe. These are somewhat arbitrary distinctions. Boundary lines between classes of machines get more blurred by the day, but our definitions will do for purposes of discussion and illustration.

If your Macintosh is located on the same premises as the mainframe or mini, you can direct connect the two, using special hardware and cables.

To connect from a remote location, from a home-based Macintosh, for example, the mainframe has to be equipped with its own modem(s), phone line(s), and protocol so that your Macintosh can dial in.

Before attempting a mainframe link, it is essential that you contact the system operator or MIS (Management Information System) manager of the host machine to get the necessary clearances and find out what the protocol requirements are. (See Chapter Eight for more about "protocols.")

The Macintosh used as a mainframe terminal (or terminal emulator) will perform differently than the same Macintosh linked to another Macintosh. For one thing, the keys might not function as you'd expect. For another, the screen display may be different.



Terminal Emulation

Terminals used to be plain teletypewriters (TTYs). They weren't very smart machines, nor could they handle graphics. Then mainframes came along. One terminal (keyboard) gave way to many "remote service locations" and they looked much like some of today's microcomputers, but they weren't very smart, either. Some of them could handle graphics, from rudimentary to full-blown, hi-resolution images. This capability is important in applications such as engineering design and weather mapping.

Terminals are getting smarter. In the meantime, more and more microcomputers are performing the role that terminals once did: they dial-up remote systems and provide local access to mainframe functions.

There are still many different kinds of terminals out there that have set families of standards among themselves and the various mainframes and minis with which they connect. Far from being obsolete, terminals are becoming smarter and looking more like micros every day.

When you run a terminal program such as MicroPhone on the Macintosh, the Macintosh becomes a smart text terminal. Since MicroPhone also handles VT100 emulation, it is also a pretty smart graphics terminal.

Using the Macintosh as a terminal emulator offers some advantages.

- Most remote terminals cannot store and process information after the link with the mainframe has been broken. These are called *dumb terminals*. Your Macintosh can collect information from the mainframe and further refine it using a spreadsheet program or word processor.
- Most dumb terminals have to be set up to access only a single remote mainframe or class of mainframes. Your Macintosh does not have this limitation. It can connect with a DEC machine and act like a VT100 terminal today, or connect with a local BBS and act like a TTY terminal tomorrow.

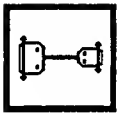
VersaTerm

A popular graphics terminal with its own set of standards is the Tektronix series (4010, 4012, 4014). Tektronix terminals are used in a wide variety of professional settings; architects, engineers, and designers of many kinds use them.

VersaTerm on the Macintosh is the package of choice for Tektronix emulation, including the Tektronix 4014. VersaTerm also works as a general purpose text terminal package. VersaTerm makes a number of different emulation packages for the Macintosh and other machines.

Common Mainframe Connections

The most common mainframes, and therefore the most likely ones you'll be using, are those from IBM. The Macintosh is capable of imitating the IBM 3270 mainframe terminal and the DEC VT100 terminal. Most of the programs covered in Chapter Five and listed in the Macintosh-to-Macintosh chart offer these two mainframe settings as part of their functions. The following notes provide additional "how-to" details for using VT100 with DEC machines.



Digital Equipment Corporation DEC VT100 To direct connect to your local DEC machine:

1. Contact the DEC system operator or administrator. Find out whether the DEC uses the VT52 escape-code protocol or the VT100 escape-code protocol (or check out what everyone else is using). Get the necessary clearances to connect directly using a properly prepared coaxial cable.
2. Connect the coaxial cable to the Macintosh modem and the communication controller port of the DEC.
3. The compatibility setting on your Macintosh terminal emulation protocol should be set to either the VT52 or the VT100 protocol, as indicated in step 1.

VT100 Emulation Digital Equipment Corporation operates an on-line "store" for its corporate customers. This system can be reached through the following number: 800 DEC DEMO. Developers who provide VT100 emulation in their terminal packages use this system to test out how well VT100 emulation actually works.

In our tests, we found that MacTerminal 2.0 and VersaTerm had the best VT100 terminal emulation. Other systems claiming VT100 emulation did not necessarily support all the enhancement features of VT100, such as underline, delete line, inverse video, and others. Many companies were actively improving their packages' VT100 performance. Here, again, it's a matter of degree. If all you need is text

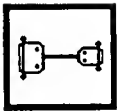
transfer support, just about any of the packages that have a VT100 emulation mode will suffice. If you need more advanced graphic support, you may want to run your own tests on a given package to make sure it supports the features you need for your application.

IBM 3270 A typical 3270 installation consists of a *cluster controller* and several terminals. The function of the cluster controller is to receive data from the mainframe computer, interpret special switching information, and then send the data to the correct terminal. In the other direction, the controller *multiplexes* (combines) the incoming data from each of the terminals and sends it through a single data line on to the 3270 for processing.

Your company may be one of many that have invested large sums in 3270 communications equipment. The terminals are called 3278s, and there may be many of these scattered throughout the company.

The specifics may vary in your case, but the following steps should guide you through the process of connecting your Macintosh to your company's 3270 system.

1. Contact the IBM mainframe's operator to get the necessary clearances to connect directly. Find out whether there is already a cluster controller installed. If not, you may want to suggest investigation of the Apple Cluster Controller hardware for use at the mainframe site. Also get the proper log-on and log-off procedures, and any required passwords.
2. Obtain *either* an AppleLine *or* the Apple Cluster Controller hardware to connect to the Macintosh modem port. Either one of these devices will perform the necessary protocol conversions to connect to the IBM. The AppleLine connects directly to existing 3274 and 3276 controllers. Again, the specifics of your case will depend on what you discover in step 1. AppleLine provides a translation between the asynchronous output of the Macintosh and the coaxial protocols of the 3270 system.
3. The necessary coaxial cable to connect the AppleLine controller or the Apple Cluster controller to the phone/modem port of the Macintosh will come with these accessories. Otherwise, you'll have to make sure you provide the right cable to connect to the IBM 3274 Cluster Controller. Again, the Apple Cluster Controller performs the functions of the IBM 3274 or 3276, except that it allows Macintoshes to connect to the system.
4. Connect the AppleLine or Cluster Controller to the IBM. In existing installations, this may already have been done. This discussion assumes use of MacTerminal 2.0, which supports 3278 emulation. Otherwise, substitute your own terminal software, but make sure it supports 3270s.



5. Make necessary setting adjustments to terminal program depending on whether you are using the AppleLine or Cluster Controller option.
6. Set bps (baud) to 9600.
7. Initialize the controller (AppleLine is assumed here—check the AppleLine controller manual) by sending the AUTOBAUD message and executing the AppleLine's Supervisor program.

Use of the Apple Cluster Controller requires either a synchronous modem or a direct cable connection. It can support up to seven asynchronous devices (Macintoshes, modems, etc.) per controller.

The connections, as you know by now, are only half the problem. The other half is in exchanging data that can be used by both the mainframe and the Macintosh. The main obstacle here depends on what software your mainframe is running. Mainframes use a variety of data base management systems (DBMS), just as micros do. Within this variety is an equally wide range of internal data structures, with different data organizations and different relationships between pieces of the information available. Each DBMS will have its own query language, as well. There are no standards for structure, request syntax, or formats for data base dictionaries (the files used by some DBMS to maintain the relationships between data sets and users). Also, a mainframe might use EBCDIC characters but store its data in hexadecimal code.



Key Behavior When it's pretending to be some kind of mainframe terminal, the Macintosh keyboard may behave differently than it does at other times. IBM and many other mainframes, for example, use a character code called EBCDIC (eb' sa dik), which stands for Expanded Binary Coded Decimal Interchange Code, rather than ASCII (see Chapter Four). Therefore, before your Macintosh can communicate with such a mainframe, its keyboard codes have to be "massaged" by your terminal emulation software before the characters are sent out.

Likewise, characters received from the mainframe will have to be translated before being displayed on your Macintosh screen. This is sometimes also called *protocol conversion*. Such conversion may also handle things such as the screen display and the command codes used to handshake between your Macintosh and the mainframe.

The Macintosh can be made to imitate any of the special function keys found on the IBM terminals. For example, MacTerminal has a pull-down menu item and special command keys that let you send out characters that can't ordinarily be generated by the Macintosh keyboard.



Macintosh Plus/MacTerminal 2.0 Key Tip

The way to send certain signals to a mainframe from a Macintosh Plus keyboard using version 2.0 of MacTerminal may not be immediately obvious to you. The following key sequences are sometimes needed:

To send:	Press:
break	Option-Enter
long break	Shift-Option-Enter
answerback message	Command-Option-Enter

Establishing a connection with your target mainframe, then, will require that you orchestrate all the variables. Ask yourself, "What do I want to do?" If you only want to exchange electronic mail and an occasional text document, the solution may be close at hand. If, on the other hand, you want to receive selected information from the mainframe already properly formatted for your Excel spreadsheet, you may have to do additional work with your local mainframe programmer or operator.



Micro to Mainframe Software Solutions

BLAST and Kermit (named after Kermit the Frog, with muppeteer Jim Henson's blessings) are protocols used to link micros and mainframes. BLAST is also the name of the only software package that runs on the Macintosh (as of this writing) that specifically incorporates the BLAST protocol. Kermit, on the other hand, is available in many Macintosh terminal programs, such as Red Ryder, VersaTerm, and inTalk.

BLAST is currently used by many major corporations on their mainframes. A partial list of users includes Control Data Corporation, Northern Telecom, Seven Up, Tymnet, United Airlines, and the US Postal Service. It is available for the following mainframes (again, a partial listing for example's sake):

Amdahl	AT&T 3B20 (Unix)	Data General MV/ALL
Harris H-series	Hewlett Packard	IBM MVS/TSO, VM/CMS
Wang VS	DEC VAX, PDP-11	

BLAST stands for **blocked asynchronous transmission**. The protocol allows the simultaneous transmission of information in *both* directions, which makes it a true full-duplex protocol. Most other protocols, such as Xmodem, don't take advantage of the full-duplex physical connection. File transfers take place in one direction at a time.

BLAST is especially useful for long-distance transmissions, through international satellite links, for example, where line noise and the need for speed usually fight with one another. It is a highly reliable error-checking protocol for eight-bit transparent transfers under adverse circumstances.

For all its technical prowess, BLAST does have some drawbacks. For one thing, if your company is not already using BLAST, your DP manager may balk at the cost of installation. For example, to install BLAST on the Amdahl or IBM mainframes, the licensing fee is \$5,500 for the mainframe package, \$250 for the Macintosh package, plus an additional fee for each different micro to be linked into the network. (A few more representative (1986) prices include: \$395 for the IBM AT running Xenix, \$895 for the DEC PDP-11, \$395 for the Macintosh XL running MacWorks.

BLAST is much less user friendly than most other packages running on the Macintosh. It doesn't take advantage of the Macintosh interface *at all*. You are presented with a command-oriented set of menus that you must maneuver your way through with the limited aid of a manual that harkens back to the bad old pre-Macintosh days. You must truly *need* the technical advantages of BLAST before you can justify working your way through its arcane structure. Again, those needs would include a combination of long-distance, highly reliable, error-free transmission, and fast transfer in both directions at once.

Kermit, on the other hand, is an equally sound error-checking protocol that is also available for many mainframe environments. It has the added advantage of being a non-proprietary protocol, unlike BLAST, which must be licensed from Communications Research Group before it can be incorporated into a software package or used on a given computer (see Chapter Eight). The protocol and its original implementations were developed at Columbia University.

Major implementations of the Kermit file transfer protocol include:

Unix	Burroughs	Cray-1
Data General	DEC VAX, PDP-11	Harris
IBM 370	PRIME	Tandem

The full listing of Kermit implementations includes over 200 machines and operating systems. Additional versions are always under development.

The Kermit software, including the source code, is furnished free, without license, and with no restrictions on copying or redistribution, except that it cannot be sold for profit and copyright notices must be left intact.

On the minus side, however, the quality of the implementations of Kermit varies from machine to machine. Some are polished, well-documented, professional products; others are not. Since the source code is provided for all implementations, users can and do make improvements in both code and documentation.

Although there is no licensing fee for Kermit packages, if you get it directly from Columbia University (which we recommend) you will have to pay a small media and handling fee. (See Chapter Eight for Kermit access information.) Source tapes and documentation, as of this writing, run in the \$100–\$200 range. The Macintosh Kermit disk and documentation cost a whopping \$15, directly from Columbia.

Kermit's primary value lies in the fact that if your mainframe computer doesn't yet support error-free mainframe-to-micro links, you can assist the update by getting a copy of the Kermit protocol to run on your mainframe for the price of the media (tape).



MacMainFrame DA

MacMainFrame is a Macintosh Desk Accessory (DA) for use with copies of MacTerminal (Apple). It is specifically designed to work with an IBM mainframe computer running a program called the Host File Transfer (HFT). Communication between the Macintosh and the mainframe is accomplished through a special black box called the Avatar PA1000 Turbo (PA1000T). This is a hardware connecting and protocol conversion device. Connections can be made either directly, if the Macintosh is in the same building as the mainframe, or through the usual modem/phone line combo. The PA1000T is attached to the IBM 3274 or 3276 Cluster Controller with a coaxial cable. This combination allows you to perform quick, vanilla text transfers in either direction. Text files are converted automatically from the mainframe formats to Macintosh formats, and vice versa. This is a handy little DA, like MacTerm (Sidekick), that is short on features but long on convenience. It is meant to work specifically with MacTerminal connected to special hardware, which, in turn, is connected to the IBM mainframe.

Upward Mobility: The AppleTalk Network

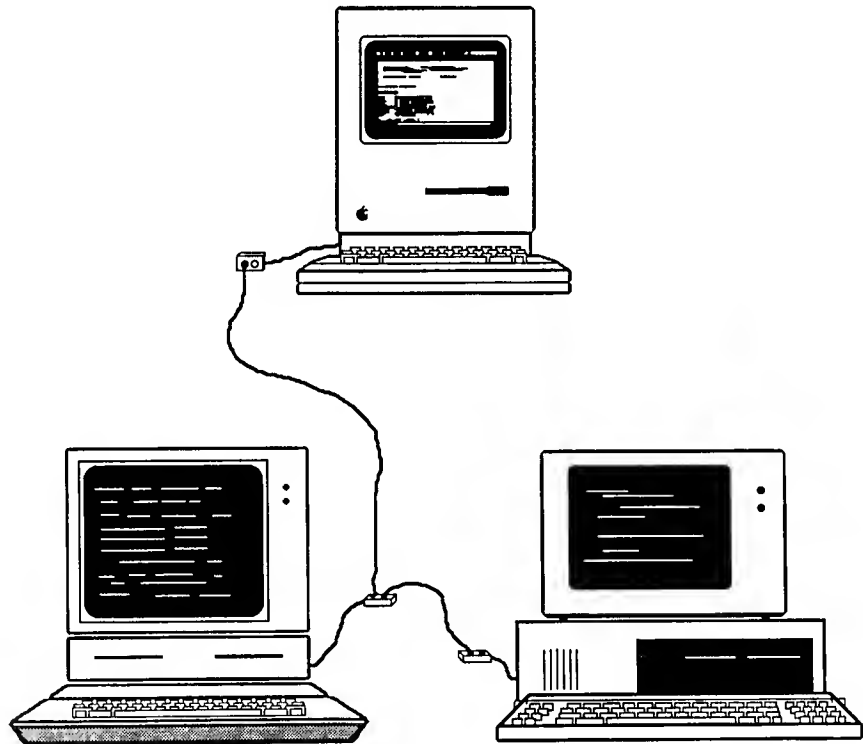


Figure 6.9 AppleTalk Network

When the day arrives that the simpler solutions we've shown you in *MacAccess* no longer suffice, and you want to permanently link your Macintosh to more than one other machine (micros or mainframes), you'll probably be ready for AppleTalk.

AppleTalk network management is built into your Macintosh. Hooking the network up couldn't be simpler. Nodes are added to the network using a low-cost cable and a variety of connection modules specific to the machines you're linking. There can be up to 32 nodes in each network. A *node* in this case is either a Macintosh, another micro, or a device such as a hard disk drive (with file server) or a LaserWriter.

The raw data transmission rate (what the line is capable of supporting exclusive of software device driver overhead and communication protocol overhead) is 230.4 kbps (230,400 bps). The linking cable can be up to 1,000 feet long. The technical name for it is a *shielded, twisted-pair cable*.



The electronics to run AppleTalk are built into your Macintosh in the form of the Zilog 8530 Serial Communications Controller. This controller can be instructed to conform to the RS-232 and the RS-422 communications protocols. AppleTalk software to run the net takes up only about 6K bytes of memory space in each machine linked to the network.

The topography of the network is known as a *multidrop cable* link. Nodes can be easily added or removed without disturbing the rest of the network. Additionally, a failing node will not interfere with the operation of the rest of the net.



TOPS

TOPS (Transcendental Operating System) by Centram Systems West, Inc., allows you to connect Apples, IBM PCs, (and Unix) machines through the AppleTalk network. TOPS acts as a distributed file server, which means that every computer on the network can use any floppy or hard disk drive on the system. From the Macintosh user's point of view, IBM PC disk drives, for example, are just like Macintosh disk drives. The IBM PC drives are represented by disk icons. The contents of the drive can be viewed by double-clicking the icon to open its window, and Macintosh files and applications can be stored on and run from that disk.

The network is completely transparent to the Macintosh user. The same is true from the IBM PC user's point of view. With TOPS, each user can control any disk drive on the network so that files can be sent or retrieved from a remote computer system even if no one is at that remote location. This is a big advantage over computer links through terminal software, wherein someone must be attending each machine, or where control is only from one side of the transaction.

There is, of course, the ultimate advantage to be had in being able to more or less permanently link up to 32 machines together, instead of just two at a time.

Note, though, that setting up TOPS and getting it running smoothly is not a trivial task. It requires careful planning and a willingness to work out the problems that are still inherent in the use of current local area network technology.

To begin with, TOPS may not be compatible with many RAM-resident PC utilities (the PC equivalent of desktop accessories). Further, the software you use on the network may not be smart enough (either at the Macintosh nodes or the PC nodes) to prevent collisions that occur

when two stations on the network try to use the same file. This has less to do with TOPS than with your application software. The result can be locked or bombed data.

Finally, with the TOPS approach, you have to make sure that individual nodes that will be in frequent use do not accidentally get turned off (in other words, taking the station off the network). Removing a computer from the TOPS network while another station is using a file from the removed computer's hard disk will have unpredictable results. The person or persons using that disk through the network may lose data and operating time. In small office environments, this may not be a problem. The larger the network, though, the more this sort of thing is a problem.

These drawbacks and questions can only begin to hint at the issues involved when you set out to install any LAN, TOPS included. Again, if the only issue for you is getting the occasional file from one machine to another, a local area network may be overkill.

Telecom Checklist

This checklist serves as a review of the elements of a successful information transfer. In any given communications situation, you may be able to skip or ignore large portions of the checklist. For example, if you will be communicating using a terminal program on the Macintosh to communicate with another system running its own copy of the same terminal program, you can skip all the items in 3, 5, and 6, since the software will take care of everything in the background.



1. What are the parameters of the source file?

Text only

Text plus formatting information

Macintosh-based

Program

Application File (e.g., spreadsheet)

2. What are the parameters of the destination file?

Text

Text plus formatting information (e.g., Word, WordStar)

Application File

Another Macintosh or ???

Used by destination system, or is destination system an intermediary? If so, will the destination system be able to handle the file "as is" or will file translations have to be done?

3. How will the transfer take place?

Phone lines and modems

Direct connect cable

If by telephone circuits, how far will the signal have to reach? Across town? Across the continent? Overseas? By satellite link? Will you be going through a connection network, such as Tymnet or Telenet? What are the local access telephone numbers, logon protocols, password requirements, account numbers, or id numbers?

4. What protocols will be used in the transfer?

Kermit

Xmodem

Ymodem

Macbinary

Vanilla ASCII with X-On/X-Off flow control

Hayes

X.25

X.PC

other?

5. If direct connect, which cable wiring spec should be used?**6. If by modem or direct connect, what other parameters will need to be set?**

bps rate (baud)

start/stop bits

data bits

protocol settings from question 4, above



Comparing Terminal Software

The following chart contains just about every feature we've found in our survey of Macintosh communications programs. The list should be treated as a jumping-off point. By highlighting those features that are absolutely essential to your needs and by adding your own features at the bottom of each section of the chart, you can use the resulting table of features to do your own comparison shopping. If you are in charge of designing a new communications system from scratch, you can use this chart and the configurations shown in this chapter to plan and cost out your design.

1.5

2

automatic

Parity

None

Odd

Even

Mark

Space

automatic

Figure 6.10 (continued)

TRANSFER PROTOCOLS

MacBinary
select receive volume
MacTerminal (Mac XModem)
XModem
1K block
CRC
variable timeout
XModem "B"
YModem
Kermit
CLINK (Crosstalk)
Telebit/DCA Fastlink
Hayes Verification
Jazz
PC to Mac and Back

1	2	3	4	5	6	1	2	3	4	1	2	3	4	1	1	2
✓	✓	✓	✓	✓	✓				✓	✓	✓	✓	✓			
✓		✓	✓	✓	✓				✓		✓	✓	✓			
✓		✓	✓	✓	✓				✓		✓	✓	✓			
✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓			
		✓			✓				✓		✓	✓	✓			
			✓	✓	✓						✓					
			✓								✓					
											✓					
✓											✓					
✓			✓													
				✓												
						✓										
							✓									

AUTOMATION

Macros
Programming language
Prewritten scripts

1	2	3	4	5	6	1	2	3	4	1	2	3	4	1	1	2
✓	✓		✓	✓	✓				✓			✓	h			
✓			✓	✓								✓				
✓	✓		✓	✓								g				

SPECIAL FEATURES

Built-in editor
Print session while on line
File format conversion/filter
Fonts
Function "keys"
HFS supported
NOT copy protected
Printer port selectable
Record session
Graphics
Security
passwords
encryption
Answer mode

1	2	3	4	5	6	1	2	3	4	1	2	3	4	1	1	2
✓	✓		✓		✓	✓	✓		✓			✓				
✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓	✓			
								✓				✓	✓			
			✓	✓		✓	✓					✓				
✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓	✓	✓		
✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓	✓	✓		
✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓	✓	✓		
g				✓	✓			✓				✓				
✓			✓		✓							✓				
✓	✓	✓	✓	✓	✓	✓	✓					✓				

SPECIAL CONNECTIONS

Mac-to-Mac
remote control
Mac-to-IBM

1	2	3	4	5	6	1	2	3	4	1	2	3	4	1	1	2
✓	✓	✓	✓	✓				✓				✓				
												✓				
c	✓		✓				✓					✓				

NOTES:

- Supports Vidtext graphics available on CompuServe.
- Accomplished using the built-in script.
- A version of inTalk is available for the IBM PC (running under Microsoft Windows) so that inTalk on the Mac can communicate with inTalk on the IBM PC. InTalk also supports Crosstalk protocol.
- Smartcom treats the Apple modem just like any other "intelligent" modem. For any modem other than the Hayes, special commands must be entered manually from the keyboard.
- DIP switches #1 and #3 must be in the down position.
- Gives static estimate of total transfer time only.
- Available on BBSs and info utilities.
- Auto logon.

Figure 6.10 (continued)



Terminal Software Summaries

The following are our summary observations about each of the programs in our chart. They are meant to give you an overall impression of the package under discussion, as well as to provide those notes that just don't fit well in a chart format. Keep in mind that many of these software packages are in a continuous state of development. Be sure to research the latest versions of any packages you are interested in (as well as any other completely new products) before you make a purchase decision.

inTalk 2.10 This is a fine telecom program. It ranks among our favorite programs, right up there with MicroPhone and MacLink. Its four levels of eight function-key macros provide quick access to whatever goodies you want to load into it. These macro keys are an integral part of inTalk's CCL. Although inTalk does not have an automatic script or application generator, as does MicroPhone, its command language is more complete, easier to debug (it displays lines of code during execution), and can be edited with a word processor, making the sharing of programs easier. It also emulates a basic set of commonly used main-frame terminals.

MacGeorge/MacMail 1.0 The main screen is an address book into which both voice and modem phone numbers (and associated addresses, if you like) can be entered. The address book is intended to be a personal contact data base. It combines modest mailing list management, including the ability to print mailing labels, with a built-in, automatic "send mail" capability. You select an address and file, set a time, and MacMail/MacGeorge will take care of the rest, assuming that the receiving computer is receptive, of course. MacGeorge or MacMail (the former is available from Racal Vadic as the Apple Macintosh Communikit and includes a cable; the latter is available from Aegis Development) offer a basic working set of terminal features, suitable for light to moderate use of on-line communication. We found set-up time to be excessive, with many barriers to the first on-line connection. It needs different modem settings and a nonconformist cable (compared with most other programs), which means that you'll have to fuss with your modem switches, and that it won't work with Apple's Personal Modem, which has no switches with which to fuss.

MacTerminal 2.0 While not the most feature-laden package on the market, Apple's MacTerminal has several things going for it. For one thing, it is thoroughly debugged, making it one of the more reliable packages available. For another, it works well with the AppleTalk network and has a complete set of VT-100 emulation features. It's relatively easy to use for newcomers.

MacTerminal is suitable for light-to-moderate users of telecom. It has no built-in macro-building facility or CCL (programming language).

MicroPhone 1.035 If we were redesigning MicroPhone from scratch, we wouldn't touch most of what's already in this package. We might add a more robust address book/directory (along the lines of MacMail) and tuck in a script command language keyword or two. And we'd probably make the scripts available for editing on a regular word processor. If this program lives up to its promise, many people are going to want to share their scripts. Right now, there is no convenient way to do so.

Smartcom II 2.2 This program is not just user friendly, it's practically entertaining, from its start-up screen fireworks to its tastefully designed icons. All the needed basic terminal features are available. Its graphics exchange mode is fun to work with, too. If we had our way, we'd beef up Smartcom II's script language and add more powerful macro capabilities.

Telescope 1.0 This is a powerful terminal package. However, before you can access all its power, you need to spend time learning a "Macro Command Language," a "Terminal Emulation Language," and a "Graphics Application Language." These, taken together, constitute Telescope's scripting capabilities. So, for example, with the Terminal Emulation Language, you can customize your terminal emulation features, fine-tuning them to work with just about any other system. Likewise, you can send graphics down the wire to be displayed immediately at the other end (which has to be a Macintosh, of course) using the Graphics Application Language (GAL). These "GAL messages" consist of character code sequences that define text sizes, fonts, and graphic shapes which the Macintosh at the other end (also running Telescope) then draws for you. Telescope claims that this is 50 times more efficient than sending the equivalent MacPaint document.

Telescape shares our “Hacker’s Heaven” award with Red Ryder. The promise is tremendous, provided you like doing a little digging and hacking yourself. Telescape is a good, all-purpose terminal with loads of extras waiting for the serious telecom user.

Jazz 1.0 The Jazz terminal is part of an integrated package containing a word processor, spreadsheet, graphics, and data base, in addition to its communication facility. Its advantage is that it is available when using Jazz. If you want to send that spreadsheet or memo you just wrote, you could conceivably save some time by using the Jazz terminal to do so. However, as a terminal package, Jazz’s telecom features are “ho-hum” at best. We’d still recommend having a full-bodied package such as MicroPhone in addition to Jazz. Certainly you should not buy Jazz on the basis of its having built-in communications. There are better reasons to get into Jazz.

PC to Mac and Back 1.0 This package falls short on several fronts. As a Macintosh-to-IBM PC terminal, it does not have some of the dedicated features you would expect, such as the ability to translate files from the Macintosh format to the PC format and back again during the process of transfer itself. Its limited file translation features must be invoked separately from the file transmission process.

On the general-purpose terminal front, PC to Mac and Back supplies generic telecommunications. It is capable of handling most of the transfers you’d encounter in everyday work, but it may balk under some conditions. For example, its Xmodem protocol is straight Xmodem, which means that there is no way to relax Xmodem’s timing so that it can be used under crowded conditions on remote services. Its lack of macros and additional protocols, such as MacBinary or Kermit, limits its usefulness.

Straight Talk 2.07 A plain, reliable package if all you want to do is communicate with one or two other systems, such as MCI Mail or Dow Jones. It lacks macros and the protocols needed by medium-to-heavy users of telecom.

VersaTerm 2.30 An industrial-strength terminal emulator, we recommend VersaTerm for anyone who has to communicate with a mainframe using graphics or any of the other features supported by various industry-standard terminals. See Appendix E for a listing of

terminals supported by Peripherals Computers & Supplies, Inc. They make several custom terminals for the Macintosh. We only looked at the DEC VT100/Tektronix package.

FreeTerm 1.8 This is yet another everyday-use terminal package. A Desk Accessory version is also available (on the Quick and Dirty Utilities package from Dreams of the Phoenix). It is free for the capturing from many information services and local bulletin board systems. There is little to find fault with here, especially considering its price. It is a stripped-down, “send and receive” package with no settings files. (Compare with Sidekick’s MacTerm DA, for example, which takes a similar approach.)

A small thing to note: when this terminal is first run, a dialog box appears asking which port you want to use. It does not default to a port that may already be in use by some other device, such as a hard disk drive. Many other packages that sail right on into a system assuming the modem port hang-up until you can reconfigure the default port settings. It is made more difficult by the fact that you have to get the program running before you can change the default settings, which means that you have to disconnect the hard disk drive . . . which means . . . It’s nice not to have to go through all this. Incidentally, Telescape also has this start-up feature.

Pretty Good Terminal 2.1 As a shareware product, Pretty Good Terminal is just what its name implies: a basic telecom package with no frills. In this it compares favorably with several higher-priced commercial offerings, such as Straight Talk. It even has a few things that some commercially produced packages don’t, such as MacBinary, MacTerminal, and Xmodem protocols (Xmodem with CRC, too!), plus function-key macros, autoanswer mode and session capture. We’d recommend this package to anyone who wants a basic, reliable, no-hassle and inexpensive package.

Red Ryder 9.4 Red Ryder is not for everyone, but it definitely has a place in the telecom universe. Like Telescape, it has a great deal of built-in power, but you have to dig and squirm to release that power. Its programming language (Chapter Five) could be more robust than it is. Since Red Ryder is constantly under revision, you’d do well to register and pay for updates for the package, not that other packages are not also constantly under revision, mind you. It’s just that Red Ryder’s

users are *all* “beta test sites” and this is part of what makes the program, author, and community of users unique in the telecom arena.

Like MicroPhone, Red Ryder has a script-creation mode that will record an on-line session for later automatic playback. This makes it somewhat easier to get into its built-in script language.

Termworks 1.3 This package doesn’t hack it when compared with other shareware terminals. The version we tested does not have MacBinary, for example. In the shareware category, we’d recommend FreeTerm or Pretty Good Terminal, instead. Check for later version numbers, though. It’s possible that this program has since been improved.

MacTerm (Sidekick) 1.10B A Desk Accessory terminal. We’d recommend this as a good “standby” communication manager. Along with the rest of Sidekick’s phone and time management features, MacTerm can give you substantial control over your telecom cost-accounting (Chapter Seven). It is strictly for generic telecom, however.

CHAPTER

7

Telephone Management

Never expect everything of anybody.

Arlen Wilson



Over the long haul, *telephone management* becomes *telecom management* and includes:

- Acquiring voice phones
- Acquiring answering machines
- Selecting long-distance carriers
- Selecting data communication carriers
- Pinpointing on-line information sources
- Cost comparison and accounting for each of the above

Furthermore, if you're part of a growing small business, you may be faced with the responsibility for upgrading your in-house telephone system, with its PBXs (Private Branch Exchange) and voice/data LANs (Local Area Networks).

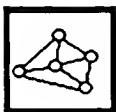
In all likelihood, more and more of your work will involve the telephone and your Macintosh. As with most other media, once you discover telecom, you find more ways to use it. For example, combining electronic publishing (everything from page layout and LaserWriter prep to providing on-line information) with telecom gives you a whole new way to prepare and print in-house publications, from reports to newsletters and books.

The infrastructure that makes these new personal media possible begins with the telephone system and the wires that come into your place of work. Let's review some of the basic cost components of this infrastructure in the context of good telephone management.

First, there is the cost of the basic telephone service. This cost includes installation of any needed wiring and phone jacks, purchase or lease of a voice phone, and selection of a long distance carrier.

Installation and wiring charges have increased to the point where it makes sense to consider wiring up your own extension phones. Once your basic line has been installed, it is a relatively simple matter to run your own lines and attach new phone jacks where you need them. Check your local Radio Shack or electronics supply store for the needed parts and wires.

Monthly telephone bills will reflect both voice telephone calls and computer-to-computer calls. If your computer-to-computer calls are long-distance, then you'll need to track your phone line charge in addition to any on-line charges that might accrue from the use of a remote service.



Special Services

Whenever you need to share Macintosh-based information with a client in another city, a basic question arises: Should you link up to a service such as MCI, which will allow you to use it as a common drop-off and pick-up point, or should you set up your machines to communicate directly with one another over the phone lines? The more direct connection (dialing the other micro directly) has many advantages over the use of an intermediate electronic mail service, such as MCI or the Source. For one thing, if all you want to do is swap files and applications with that individual or company, you'll both save money in the long run by setting up your own system. For another, there's greater security and privacy in connecting Macintosh to micro. If you share sensitive information that may be, say, associated with new software development, controlling all aspects of your data communications decreases the chance of electronic "leaks" and eavesdropping. Maintaining your own electronic storage (on-line services charge you for storing your files on their systems) may also make better economic and security sense.

If, on the other hand, you must coordinate information between several participants in different parts of the country, it might be better to use a common service provider to share your information.

Under some circumstances it's convenient to have at least two telephone lines, one for your Macintosh and one for your voice calls. In a two- or three-person business, for example, three lines make sense: one for data, one for voice, and one for mixed use. The dedicated lines are primarily for incoming calls: give one number out for voice callers, and the other number out to data callers who will be calling to connect to your Macintosh. The mixed-use line is primarily for outgoing calls, both voice and data. If you need to connect two of your own computers through the phone lines, connect the computer initiating the call to the mixed-use line (outgoing), and connect the answering computer to the incoming data line.

Don't get the call-waiting feature on the phone lines you'll be using for telecom. The signal that lets you know a second call is coming in will interfere with any data transmission you're trying to do at the time, and it may even cause your connection to be broken altogether.



Call-Waiting Tip

If you use the same line for voice and data, you can still have Call Waiting. You just need to remember to turn off the call-waiting feature by pressing *70 on the Touch Tone pad before dialing out with your Macintosh. To restore Call Waiting again, press *73.

If you do a lot of mixed voice and data work, this arrangement will also make it easier to establish new connections via computer. It is also more flexible than having a single, integrated voice and data work station.

For example, if you are having trouble connecting with a particular system, you can call up the other party on the voice line and have him or her "talk you through" the connection procedure that you perform with your Macintosh and the computer phone line. With both you and your computer on line at the same time, you can speed up any troubleshooting that needs to be done. Incidentally, current experiments in telephone transmission will make it possible for both voice and data to use the same telephone line (through a special "black box modem" that combines the two before sending them down the wire). Until then, having two lines is definitely an asset in telecom work.

Giving your Macintosh its own phone line also makes it much easier to track the costs associated solely with your telecom work. Other telecom costs have to do with the so-called specialized carriers and information providers. Long-distance services such as MCI and US

Sprint are *specialized carriers*. Services specially oriented to data communications, such as Telenet, also fall in this category.

Information providers or data base services or information utilities all charge some combination of connect charge (for the amount of time spent using the service) and information charge (a fee added to your bill to pay for specific information that is billed beyond the basic connect charges). Services such as Dialog or Newsnet, for example, charge varying rates for the different kinds of information accessed.

Telephone management becomes even more of a necessity if you are doing work for and with clients on line. If you are exchanging information with a client on line, you will need to track the telephone costs associated with the job in order to make sure your fees reflect the real costs incurred.

Alternative Long Distance

Until recently, consumers were advised to shop around for alternatives to AT&T's long-distance telephone service. However, due to FCC and court rulings, differences in costs between the various services have evened out. The other consideration—line quality—has also evened out. You get more or less the same transmission quality today no matter which service you're using. (On any given day, even AT&T may suffer noise bursts and static on the line.)

There are now six major long-distance carriers: AT&T, Allnet, MCI, US Sprint, ITT, and Western Union. While you must choose the service you want under equal access rules (if you don't, your local phone company will assign one to you), you needn't stick with this service for every call or for any specific length of time. Within the first six months of choosing a service, you can change companies free. After that, if you get dissatisfied or find a better rate, you can change for a small charge (\$5 or so in most localities).

If you make a lot of long-distance calls, you may want to analyze your bill to see if changing services will reduce your costs. To do that, you need to take into account a number of variables: the places you call frequently, the times at which you make most of your calls, the rates in effect at those times, and the amount of time spent on line.

Phone Bills

You need not analyze your own phone bill. Just send one of your typical bills to Consumer's Checkbook in Washington, DC. They will do it for you and charge you for it. Since the analysis may pay off in reduced

rates over the long haul, the charge may be worth it, or it may be a deductible business expense. Analysis charges range from \$10 for a \$10 or less phone bill, to a high of \$75 for a bill in the \$200 or greater range. For more information, call toll free 800-441-8933.

You can use any of the alternative phone services at any time simply by dialing its access number when you make your call. The access numbers are listed here, along with the toll-free 800 numbers you can call for more information.

Service	Access Code	Information Service
Western Union	10220	800-562-0240
US Sprint	10777	800-521-4949
MCI	10222	(check local listing)
ITT	10488	800-526-3000
AT&T	10288	800-222-0300
Allnet	10444	800-982-8888

Finally, some of the companies offer volume discounts which may help you if your monthly bill includes more than \$20 for long-distance charges. However, before you can take advantage of such discounts, you must choose the carrier and use it all the time, or, alternatively, set up a separate account with the carrier, as you had to do before equal access was instituted. Volume discounts will not apply if you use the five-digit codes listed above.

On-Line Logs

How you set up your on-line log will depend on the type of services you render. Client billing may require that you log each voice and each data call. You may also need to bill separately for each account you set up on a major service, such as MCI Mail or CompuServe.

Message management will require that you keep track of electronic-mail accounts and the passwords and log-on procedures associated with each.

Sideline Scenario

You have a sideline business consulting with people who want you to help design their Excel and Multiplan spreadsheets. Much of your work is done by phone, and thus you charge for that time. In order to charge fairly, you need to set realistic rates and take into account whether you

placed the call or the client did. In addition, you may need to record the following for each call:

- The date
- The time of day the call was placed
- Who originated the call
- Who the client or account is
- Whether it was a voice call or a data call
- The total elapsed time you spent on the phone
- Subject matter of the conversation or data exchange
- Any additional notes you want to remember
- The billing rate for this client and this particular project

How you organize this information is a matter of personal preference. Having such records, though, can mean the difference between staying profitable and losing a little or a lot of money on each transaction.



File Transfer Times

Table 7.1 can give you a rough estimate of on-line times required to transfer documents in the given size ranges. The transfer time, in minutes, assumes uninterrupted transfers. Capturing similar-sized files from a busy system with many simultaneous users could take considerably longer, especially if the lines are noisy and an error-checking protocol is being used.

Table 7.1 Approximate Times, in Minutes, for File Transfers

File Size(K)	Baud		
	300	1200	2400
5	3.0	.75	.37
10	6.0	1.5	.75
15	9.0	2.25	1.10
20	12.0	3.0	1.5
30	18.0	4.5	2.0
50	30.0	7.5	3.75
100	60.0	15.0	7.5



Telecom Techniques in the Spirit of the One-Minute Manager

- Spend a minute getting to know your terminal software each time you use it. The better you know how your particular software and Macintosh combination works, the further you can push it. It pays to squeeze all you can out of your terminal package.
- Print out the help files of any systems you may use frequently, or get their manuals if they're available. This is especially important if you use many systems and need to find information fast while on line. Scrolling through and trying to read reams of on-line help files can be costly. Better to do an on-line scan once, at the beginning, and have the files available for perusal in your off-line minutes.
- Compose your on-line replies in a minute or less. A primary tenet of on-line communication: be as brief as you can, always. This means sending responses that contain only the information absolutely needed to get the job done. It means getting your mail and composing your longer responses while off line.
- Try to remember that the people who will be getting your electronic mail probably are experiencing as much of an "info-overload" as you are. Compose your message so that the person who gets it can dispose of any responsibility quickly and gracefully. Short, single-action-item memos are better than a lengthy progress report with many questions and action items.
- Don't underestimate the value of a pencil and paper. Keep these ordinary tools next to your Macintosh so you can jot down thoughts that occur to you while you're connected to a service or you can make notes of things that you want to follow up on later. Sure, you can use your Notepad (on the Macintosh) to take down these notes, but sometimes that won't be convenient. You may want to remember the names or account numbers of people whose ideas have inspired you.
- Keep a data book with a section for each of the major services or accounts you frequent. (See Figure 7.1 for a sample data-book page.) Keep your captured hard copy of help files in the data book, with dividers, for ready reference. You can keep copies of your on-line exchanges or sessions in the data book, too. Alternatively, you may want to set up regular file folders for keeping archival copies of such information.
- If you really can't stand paper in any form, you can create a separate disk for storing all or some of the information we suggest here. Then you have to be conscientious about maintaining your disks and a disk file catalog. This strategy has the added advantage of automatically keeping a chronological record of your on-line activities.

- You may want to keep a file of on-line tricks you learn while using services such as CompuServe.
- Make a list of files and information you want before you log on to a remote service. Use one session just to search and make notes of indexes of files.
- Your data book can be a convenient repository for the information you'll need while traveling. You don't want to have to carry all those manuals with you, do you? Record only the relevant information for travel purposes.

Sample Quick-Reference Sheet The form shown in Figure 7.1 can be adapted to your own situation. We recommend keeping information about each system you connect with frequently on a quick-reference sheet such as this one. It can serve as a backup information source for items that are normally stored inside your communications settings files (see Chapter Four).

On Being Prepared

Much has been written about *information search specialists* who, for a fee, will research your technical, scientific, or economic questions using on-line data bases and their own computers.

Essentially, what happens is this: You submit a research question or a list of questions that you want answered. The search specialist will then conduct an on-line search of one or more of the many data bases that are now available. Using a terminal or, more likely, a microcomputer, the searcher dials up a data base to search. Then, using the search commands required by the data base, the specialist will zero in on and capture the relevant information. Most often, you'll get back not the actual information you're looking for but, rather, a list of articles, books, or university publications in which you can find the data you're seeking.

Aside from the cost savings you may realize by doing your own dial-up research, you may benefit more from doing it yourself. For one thing, you probably have a better grasp of the overall context in which your question arises than any professional data broker will have. For another, you always learn more about your own field each time you set out to answer a specific question that you need to have answered.

In such cases, part of good telecommunications management will include doing your homework *before* you connect with the service you're going to search.

Telecom Reference Sheet

NAME: _____
 ID#: _____
 PASSWORD: _____

[Note that if you keep your password here, safeguard this sheet just as you would your credit card.]

Terminal Programming Information

Duplex [Full/Half]:	_____	# Columns [80/132]:	_____
Word length [7/8 bits]:	_____	X-On/X-Off [Y/N]:	_____
Parity [N/O/E/M/S]:	_____	Protocols:	_____
BPS [2400/1200/300]:	_____	MacBinary	Xmodem Kermit Other
Stop bits [1/1.5/2]:	_____		

Phone Numbers

Direct connect: (____) _____-	_____			
Local Data Network: _____-	_____	Tymnet	Xmodem Kermit Other	

Log-on Sequence

[Sample. Your responses shown in *italics*.
 <cr> indicates carriage return.]

CONNECT <cr> <cr>
 Terminal: <cr>
 @ C 21566 <cr>
 ID: 3128,49382 <cr>
 PASSWORD? *stupid/phrase* <cr>

Command Summary

R - read	[Samples shown here.]
B - browse	
I - index	
C - complete	
S - starting	

Notes

Figure 7.1 Quick-Reference Sheet

Rather than waiting until you are connected to Dialog, for example, with its hundreds of data bases to search, and then doing your search solely by trial and error, a little preconnect-time work will help you with the search and cut down on your connect charges.

Before you begin, jot down a preliminary *search statement*. For example, suppose you are trying to decide whether to write new Macintosh software based on the availability of 2400 bps modems. One of the things you might want to find out is what the potential market for such software will be. In addition to knowing how many Macintoshes there are, you need to find out what the 2400 bps modem market will look like in two years, which is how long your software development project will take. Your search statement, then, might look something like this:

What is the estimated growth of 2400 bps modems over the next five years?

The search statement helps you identify what you're after. Here is that statement with the key words highlighted:

What is the estimated *growth* of *2400 bps modems* over the next *five years*?

You can expand or narrow your search based on any of these keywords. You can look for references or articles to any one or a combination of these keywords.

If you used just *2400 bps modems*, your search would likely turn up thousands of entries. By modifying your search with the keywords *growth* and *five years*, your search is narrowed. As you get responses, you can further refine your search statement. In some cases, you may have to refine your keywords. For example, instead of five years, as above, you might refine your keyword to specific target years (through 1990) or a range of years (1987–1995).

As critical as a good search statement is, an equally critical factor is the knowledge you bring to the search. If you are an expert in electronics industry stocks, for example, you wouldn't want to send just anyone off searching for information about your specialty. Various market reports on particular stock issues may have very different projections about how well the companies will do. Your knowledge of the industry can help you decide which information is on target for your purposes. As always, there is no substitute for your own judgment when it comes to deciding what is critical information and what is just plain data.

You should also be aware of two other facts about on-line services: on one hand, they often have information in them that is not available in print; on the other hand, they often do not have information that has already appeared in print, sometimes months ago. It can take up to several months, for example, for some data bases to be updated after a journal or article has appeared, so you are not *necessarily* getting up-to-the-minute or state-of-the-art information just because you got it from an on-line data base. It's usually a good idea to back up your on-line research with trips to your local library.



Software for Telecom Management

In general-purpose terminal programs, two features are helpful for good telecom management:

1. It's useful to be able to keep a log, like the one in Figure 7.2, of elapsed time or connect time, so that you can compare your connect-time records with your monthly bill. Computers are run by people. In combination they have been known to make mistakes on telephone bills and on-line service bills.
2. It's useful to be able to set up your terminal program to send and receive mail during off-peak hours in order to save on both connect charges and long-distance telephone charges, if any.

MicroPhone and inTalk both provide a wait feature that will delay dialing until a time that you specify. By using that feature in your own custom scripts, you can reduce access charges considerably. (See Chapter Five for more about custom programming telecom packages.)

Sidekick

Sidekick (Figure 7.3) is a software package that provides a popular set of desk accessories for the Macintosh (from Borland International—see Appendix C). It has a phone directory in which you can keep the names and phone numbers of clients, co-workers, and friends. When you place a voice call or when you receive an incoming voice call from one of these individuals, you can record the call in Sidekick's phone log, shown in Figure 7.4. You then have a dated record of the length of the call and any notes about the call that you want to keep.

MacGEORGE™ File Edit Mail Phone Modem Labels Options Help					
Transfer Log					
Who From/To	Start Date/Time		End Date/Time		Status
Smith, John	5/17/86	9:38 AM	5/17/86	9:39 AM	(CS) Aborted Here
Cables: "Mac Plus Adapter Cable"					
Smith, John	5/17/86	9:41 AM	5/17/86	9:43 AM	(CS) Transfer Errors
Cables: "Mac Plus Adapter Cable"					
Smith, John	5/17/86	9:46 AM	5/17/86	9:49 AM	(CS) Complete Send
Cables: "MacLink Cable"					
	5/17/86	9:59 AM	5/17/86	9:59 AM	(CA) Auto Received
MacMail: Rydberg					
TREE	05/27/86	11:38 PM	-----	-----	(CS) To be sent.
MacGeorge:Clipboard File					

MacMail/MacGeorge keeps a log
of your messages and their status

Figure 7.2 MacMail/MacGeorge Transfer Log

Sidekick keeps track of your connect time and calculates phone charges and consulting fees

You can type notes in here

Figure 7.3 Sidekick

Phone Log			
Phone Number-- Name--	Start-- Charges--	Stop-- Cons!tg.--	Length-- Date--
1(800)446-3000 Apple Computer Called to get same information about the Macintosh computer.	9:57:00 AM \$0.96	9:57:58 AM \$0.16	00:00:58 Mon, Aug 19, 1985
1(408)438-8400 Barland International This is an example of phone notes taken while talking on the phone. Using Sidekick for the Macintosh from Barland International is easy.	10:02:59 AM \$0.54	10:04:04 AM \$0.09	00:01:05 Mon, Aug 19, 1985
1(408)996-1010 Apple Computer	10:58:54 AM \$0.06	10:58:58 AM \$0.01	00:00:04 Fri, Jan 17, 1986
1(23)456-7890 Gengle, Dean	4:39:58 PM \$0.00	4:40:05 PM \$0.06	00:00:07 Sat, Mar 15, 1986
1(908)765-4321 Cortland, Joe	4:45:12 PM \$0.00	4:45:24 PM \$0.00	00:00:12 Sat, Mar 15, 1986
531-3700 CIS	5:18:05 PM \$0.00	5:22:13 PM \$0.00	00:04:08 Sat, Mar 15, 1986
NOTES: Sidekick Test Connection			
531-3700 CIS	5:45:17 PM \$2.25	5:45:26 PM \$0.00	00:00:09 Sat, Mar 15, 1986
	5:46:05 PM \$0.00	5:47:16 PM \$0.00	00:01:11 Sat, Mar 15, 1986

Sidekick creates a log of your calls (including connect time, phone charges, and consulting fees) in a text file

Figure 7.4 Sidekick Log



Disk Storage and Management

You may find it convenient to keep your system notes and communications disks in an 8-1/2-by-11-inch notebook. Photo display/storage sleeves with three holes punched in them can double as disk holders. Each page accommodates six 3.5-inch Macintosh disks. (This is the same size as prints from 126 film.) Filler sleeves are available at your local photo processor, camera store, or by mail from:

20th Century Plastics (order Model #WH-86)

P.O. Box 51003

Los Angeles, CA 90051



The Business Communications Crisis

Anyone in business *must* communicate information: to reserve hotel rooms, rent cars, confirm prices or deliveries, provide decision-making data, and so forth. That much all businesses and the providers of business communications services agree on: nothing can happen until the right information reaches its destination at the right time.

More and more, businesspeople are realizing that their markets are global, not just regional or national. As that happens, the services of international record carriers and packet-switching data networks become more important. Even within the continental U.S., however, the providers of network services have been slow to invest in computers and support equipment that would simplify the movement of data streams from one location to another.

Instead, customers have been forced to meet outdated requirements and supply their own solutions to incompatibility problems. The proliferation of telecom speeds, transfer protocols, languages, and interface equipment (modems, multiplexors, protocol conversion "black boxes" and all the rest) has meant capital outlay for users, while providers have spent their dollars on public relations and marketing campaigns, trying to tell us how great the networked nation is going to be . . . someday.

Today, a business may use as much as 20% of its information processing dollar on communications overhead. This overhead includes many of the things we've covered in *MacAccess*: file preparation, file translation, protocol conversion, and so on. In addition, the average business organization has to work with half a dozen to a dozen different communications systems to reach all its branch offices, suppliers, overseas offices, service representatives, and customers. (A representative list: AT&T, US Sprint, MCI, MCI Mail, U.S. Mail (paper), Telex, TWX, facsimile, CompuServe, videocassettes, paper-based newsletters.)

It is not that our communications networks are poorly designed or maintained. On the contrary, there are so many new technologies, products, services and combinations of services on the scene that the average non-specialist sees the wealth of possibilities as a quagmire of conflicting claims. Somehow, these new services and devices have to work together, and the only ones who can make a given combination work together, today, are the customers.

There are now battalions of consultants who ostensibly help organizations spend some of that 20% of communications dollars (overhead) we alluded to earlier. We recommend that companies develop their own in-house expertise and training programs in all aspects of communication, from desktop publishing to electronic publishing to electronic mail. Groom an information resource manager to work with your DP department and your communications people. Remember that every choice you make in your mix of communications represents

a commitment of capital and may lock you into old technologies (or out of new ones). The next-best thing to in-house expertise is a good on-going relationship with consulting telecommunications specialists.

Eventually, this crisis will be resolved. Network services—the people who actually are responsible for the connections between machines—are slowly improving their networks by incorporating more intelligent protocols that can tailor themselves to your terminal. They will eventually “know” your computer and will accept files from you any way you want to send them, and they will make sure that they arrive at the other end in a form that can be used. A good current example is MCI Mail, which allows a wide variety of exchanges between users to take place transparently and with little or no hassle.

As one businessman put it, “Forcing users to become communications specialists absorbs time and money, a luxury no longer affordable when data communications is as vital as telephone calls.”

CHAPTER

8

Advanced Technical Topics

*He who looks for a mule without fault must
go on foot.*

Spanish Proverb

As you integrate telecom into your softspace, you'll find your learning curve becoming less steep. The routine matters of communicating in this new way will be more automatic. You'll be able to put more of your conscious attention into your real work, whatever that is.

In this chapter you'll find advanced information that we think may help you understand more of the terminology, technical issues, and practical work of Macintosh telecommunications.

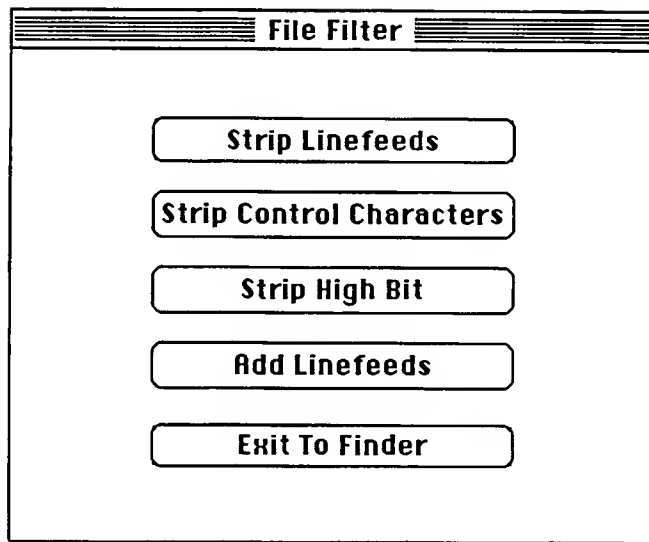
Today, telecom is no longer in its infancy, certainly, but it's not yet fully grown either. At this somewhat awkward in-between stage, where there are competing standards, where the technology is changing and new technology is being invented every hour, where the surface of possibilities has barely been scratched, you'd do well to keep a notebook. You can also use the margins of these pages to add your own notes and amendments to *MacAccess*.

Each major topic that follows in this chapter is meant to stand on its own, so let your interests and needs take you where they will.



Files and File Formats

When you receive a text file from another non-Macintosh computer, you may not always get it in a condition that is immediately useable. If



File Filter is a public domain utility available on your neighborhood on-line service or the BMUG (see text)

Figure 8.2 File Filter



File Conversion and Exchange Formats: DIF and SYLK

DIF stands for Data Interchange Format. It was developed by the makers of VisiCalc to facilitate the exchange of spreadsheet data between different programs running on different machines. Many spreadsheet and data base programs will import and export files in the DIF format. Incidentally, the internal VisiCalc file format is sometimes referred to as the *VC format*. It is *not* the same as DIF.

SYLK stands for the SYmbolic LinK format. SYLK was developed by Microsoft, for the same reason: to be able to exchange files between different kinds of programs. It provides for data compression (which DIF does not) as well as complete file formats.

If a given program, such as Multiplan, runs on both the Macintosh and on the target machine to which you'll be sending a Multiplan file, you needn't use an intermediate file-exchange format such as SYLK. All that's necessary is that you send the file to the target machine using either a MacBinary or Xmodem transfer. Assuming that the other machine will be running its version of the same program, it will be able to use the transmitted information as is.

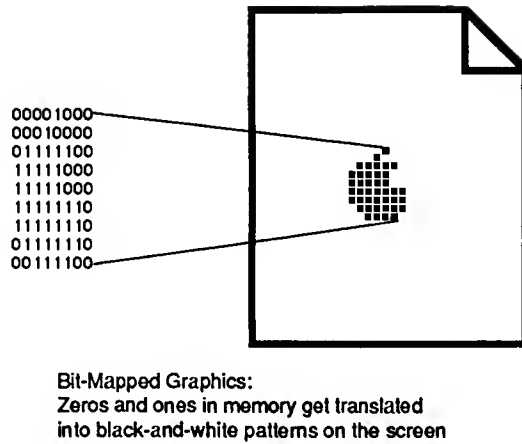


Figure 8.3 Bit-Mapped Graphics

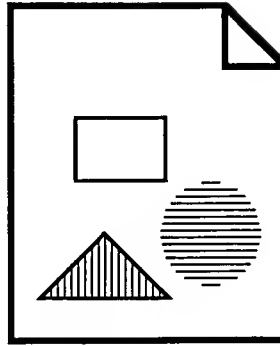
For more information on this topic, see:

- *Software Arts Technical Notes*, SATN, 18, FATN, "Programmer's Guide to Data Interchange Formats."
- *Microsoft Multiplan Manual*, Appendix 4, "The SYLK 'Symbolic Link' File Format."

Macintosh File Data Types

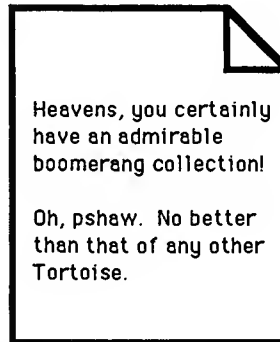
Macintosh files contain one of the following four basic types of information in their data forks.

1. Bit-mapped graphics (Figure 8.3). These are simply patterns of black and white dots represented in memory by ones and zeros, respectively. Bit-mapped graphics are manipulated on the bit level by changing the state of a bit from one to zero or back again.
2. QuickDraw graphics (Figure 8.4). QuickDraw graphics are one step more "intelligent" than bit-mapped graphics. They are made up of geometric shapes represented in the Macintosh Toolbox (ROM) by mathematical formulas. The various shapes retain their integrity and can be moved and otherwise manipulated, depending on the application program that uses them.
3. Text (Figure 8.5). The familiar ASCII code files used by word processors on the Macintosh are enhanced by font size and style information. There are



QuickDraw Graphics:
Geometric objects are drawn on the screen
and can be manipulated

Figure 8.4 QuickDraw Graphics

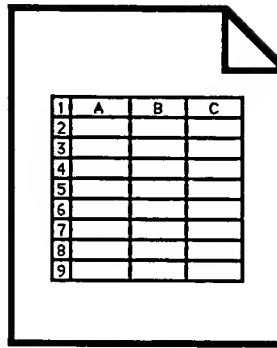


Text Info:
ASCII code used by most applications

Figure 8.5 Text

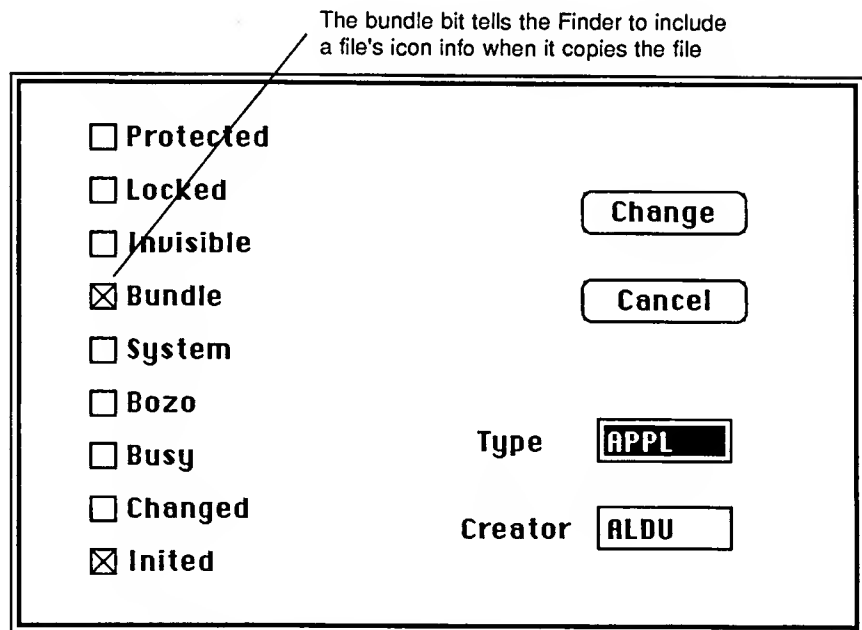
many “subspecies” of text files. Vanilla text can usually be created by choosing the Text option when saving a file from within a given application program.

4. Cellular (Figure 8.6). Spreadsheet and data base information is stored in cellular form. Words and numbers are kept in blocks that are delineated by tab characters.



Cellular Info:
Cells are composed of words and numbers.
In a file the cell blocks are separated by tabs

Figure 8.6 Cellular Info



Use programs like Fedit to modify a file's
header information

Figure 8.7 Bundle Bit



Bundle Bits

When you copy a program from one Macintosh disk to another or attempt to send it down the wires to another Macintosh, all of the resources stored in the resource fork (see Chapter Three) of the file that the Finder expects must also accompany the file if it's to be useful. Finder resources include the icon information for the application and the documents produced by the application. Each program has an on-off flag called a *bundle bit* (Figure 8.7) that tells the Finder whether any such resources are present and need to be copied. If the bundle bit is set, the Finder will copy the program's bundle resource and bring along the needed information. If the bundle bit isn't set, the program's icons will get left behind. Some terminal programs may "forget" to transmit this information. You can use a program such as Fedit to set the bundle bit yourself, thus assuring all the program's finder info will be sent.



Using BinHex

If you don't have a general-purpose terminal program that supports MacBinary, you will not be able to transfer and use Macintosh programs directly. ASCII transfers will strip out the icons, application identifiers, and other information that you need to run a Macintosh program.

BinHex is the name of a conversion program you run on your Macintosh specifically to solve this problem. It predates the MacBinary solution and has gone through many metamorphoses to its current stable version.

BinHex simply converts outgoing files for transmission to another non-Macintosh system, and it reconverts files that you capture back into MacIntelligible information. That's all it does.

The BinHex conversion treats a Macintosh file as just another stream of bits. BinHex takes this ongoing flow of bits, counts them, and puts them into sets (blocks) of six bits each. Then, each block of six bits is converted into one of 64 ASCII characters ($2 \text{ to the } 6\text{th power} = 64$) shown here:

```
!"#$%&'()*+,-012345689@ABCDEFGHIJKLMNPQRSTUVWXYZ[`abcdefghijklmnopqr
```

These characters were (allegedly) chosen for maximum noise protection. Each line in a BinHexed file starts and ends with a : (colon).

There is a maximum of 64 characters on a line. A BinHexed file can also be preceded by a comment or comments. Thus, if you look at such a file with a word processor, you'll be able to see the comments. The comments are stripped out of the reconstituted Macintosh file. For example, when you look at a file that has not yet been reconverted from a BinHex to a Macintosh-useable file, you may see one or more of the following lines:

(This file must be converted with BinHex.Hex)

(This file must be converted with BinHex 4.0)

\$APPLBOZ\$2000

***COMPRESSED

***RESOURCE FORK

When browsing the files on remote computers, you can usually tell whether or not a given file has been BinHexed because it will have a file extension such as .HQX, .HEX, .HCX, or .BIN.

When capturing a BinHex file from another system, you can use any transfer method: straight ASCII (text), Xmodem, and so on. Many computer systems contain useful files that were contributed by users who didn't have MacBinary. They had to use BinHex to send their files to these systems, so there are still a lot of useful free Macintosh programs around in BinHex files. Once you get the file, though, you'll have to convert it before you can use it. BinHex 5.0 should be in your box of telecom tricks.



The Evolution of a Meme: BinHex 5.0

The curious reader might well ask "Why are there so many different extensions? I mean, really . . . h-q-x, h-c-x, and even *bin*! Why not just settle on one and agree? There must be a reason for this!"

(A "meme" is a unit of cultural information. Memes are to cultural and technical evolution what genes are to biological evolution.)

To the best of our knowledge, here it is. The most common BinHex file extension you'll find out there is .BIN. The rest of the different extensions have arisen as a consequence of BinHex's evolution.

BinHex began life as an idea meant to solve a problem. Then it became a BASIC program. As a BASIC program, BinHex got all the way up to version 3.0 before it became generally available to the Macintosh public. BinHex 3.0 took 20 to 30 minutes to convert or reconvert a Macintosh file.

Then, for a brief time, a faster, compiled BASIC version of BinHex made the rounds. This version improved the speed, but it still wasn't optimal.

When the same conversion approach was written in assembly language, it became BinHex 1.0 to distinguish it from the earlier, slower BASIC versions.

BinHex 2.0 was the second round of the assembly language version. It created compressed files so it made files with an .HCX (C for compressed) extension in order to show that you needed to have BinHex 2.0, and not an earlier version, in order to convert the file back into Macintosh-readable form (and expand the compressed information). BinHex 2.0 was pretty smart. It was able to read and reconvert files created with both the earlier BASIC version and BinHex 1.0. Both these versions, incidentally, create their converted files with an .HEX extension.

Months later (are you getting all this?) BinHex 4.0 emerged. True to the evolutionary line, this was also an assembly language incarnation. It became version 4.0 so that it could not be confused with the earlier, BASIC BinHex 3.0. It used a slightly different conversion protocol, so a new file extension was needed: .HQX. BinHex 4.0 could understand and convert both .HEX and .HCX converted files, but only *produced* .HQX files.

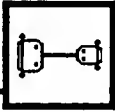
Then the MacBinary protocol emerged. BinHex 5.0 is MacBinary compatible. Let's hope it's the last in the series. It will, according to Yves Lempereur, understand any previously created BinHex file, whether .HQX, .HEX, .HCX, or just plain .BIN. It only creates MacBinary files, however.

If you happen to run across one of the earlier versions of BinHex try to stamp it out. We think that BinHex 5.0 is the most up-to-date and has been taken as far as it can go before it becomes extinct altogether.



How to Get BinHex

We have provided a copy of BinHex 5.0 in Appendix E in a Softstrip. You only need to read this strip into your Macintosh to get your working copy of BinHex 5.0. If you do not have a Softstrip reader, BinHex 5.0 is also available on many information services, such as CompuServe, and on local Macintosh-oriented special interest networks. You can also get a copy direct from the author, Yves Lempereur. (See Appendix E for ordering information.)



Cables

The diagrams and pin-out descriptions on the following pages are offered as a guideline to your own cabling efforts, if you should happen to need them. Often, two different cables with two slightly different wiring schemes will work in a given situation.

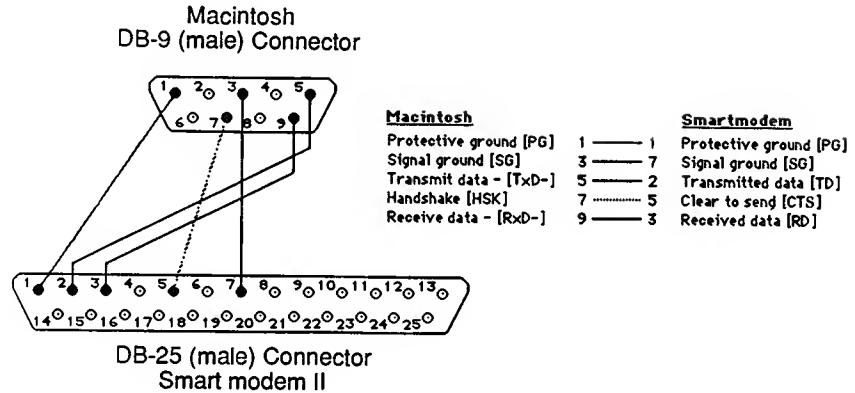
CAUTION: The diagrams presented here are for configurations that we know to work in the given situations. If you feel uncertain about the whole matter, turn our diagrams and your cabling problem over to a competent technician who can make the cable for you.

When all else fails, experiment. The worst that can happen is nothing. When you do experiment, keep careful records, just as any scientist would, because that's what you are when you experiment: an information scientist testing the empirical world to see what's going on. Your notebook should record those instances where you tried a particular wiring scheme or cable and found that it didn't work. Remember that telecom is a peculiar mixture of the exact and the inexact.

As you will see, handshaking is often optional. To begin with, try connecting just the ground, receive data, and transmit data lines and see if that works.

All cables may be classed as either *straight modem* or *null-modem* types. The straight types connect the RD (Receive Data) pin on one side to the RD pin on the other, and the TD (Transmit Data) pin on one side to the corresponding TD pin on the other. Null-modem cables cross-connect the RD to the TD, and the TD to the RD.

There are different ways to present cable diagrams. In *MacAccess* we labeled the pins on each connector in accordance with the device they plug into. Other sources may label only the *connection* between the pins. Where we have *TD* pin to *TD* pin, they might have the *Transmit Data* line. Each way has its advantages. Ours is that it should make it a little easier to actually get in there and connect it all up yourself.



NOTES:

Every manual provided a slightly different variation of the cable connection between the Macintosh and the Hayes Smartmodem, proving that cabling is not an exact science. We've provided the cabling recommended by Hayes themselves. Note that they recommend connecting the protective ground pins [PG] but the connection 7-5 (handshaking) is optional.

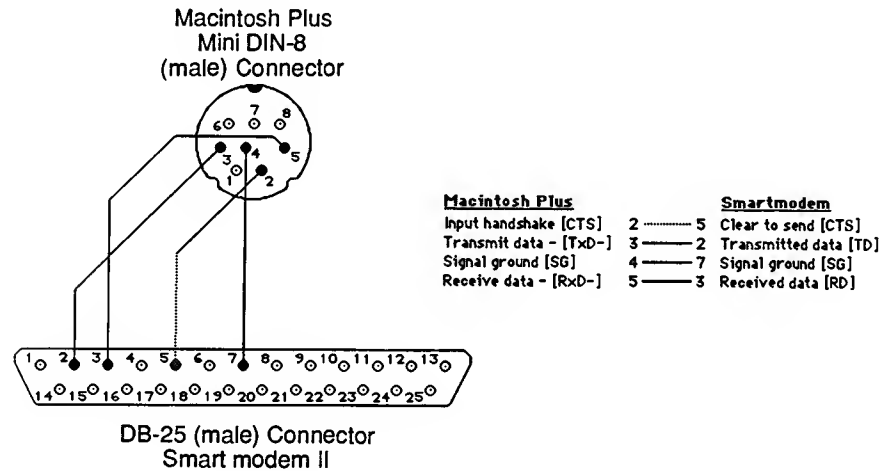
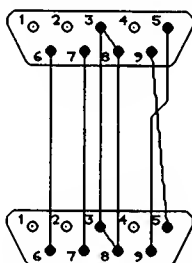


Figure 8.8a Smartmodem Cable

Macintosh
DB-9 (male) Connector



Macintosh

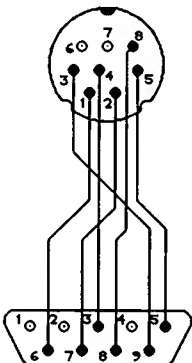
Signal ground [SG]
Transmit data - [TxD-]
+12 volts
Handshake [HSK]
Receive data + [RxD+]
Receive data - [RxD-]

Apple Modem

Signal ground [SG]
Transmit data [TxD]
Data terminal ready [DTR]
Data carrier detect [DCD]
Protective ground [PG]
Receive data [RxD]

DB-9 (male) Connector
Apple Modem

Macintosh Plus
Mini DIN-8
(male) Connector



Macintosh Plus

Output handshake (DTR)
Input handshake (CTS)
Transmit data - [TxD-]
Signal ground [SG]
Receive data - [RxD-]
Receive data + [RxD+]

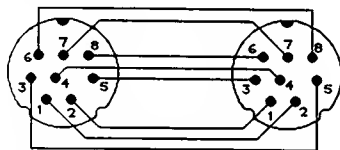
Apple Modem

Data terminal ready [DTR]
Data carrier detect [CDC]
Transmit data [TXD]
Signal ground [SGND]
Receive data [RCD]
Chassis ground [GND]

DB-9 (male) Connector
Apple Modem

Figure 8.8b Apple Modem Cable

Mini DIN-8
(male) Connector Mini DIN-8
(male) Connector



Macintosh Plus Apple Personal Modem/
Imagewriter II

Macintosh Plus

Output handshake [DTR]
Input handshake [CTS]
Transmit data - [TxD-]
Signal ground [SG]
Receive data - [RxD-]
Transmit data + [TxD+]
Receive data + [RxD+]

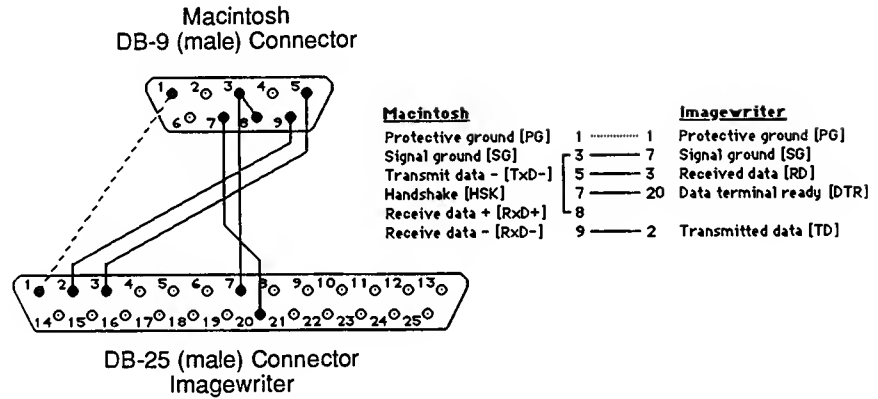
Imagewriter/ Personal Modem

1 — 2 Data terminal ready [DTR]
2 — 1 Data set ready [DSR]
3 — 5 Transmit data [TxD]
4 — 4 Signal ground [SG]
5 — 3 Receive data [RxD]
6 — 8 not connected
7 — 7 Data carrier detect [DCD]
8 — 6 Signal ground [SG]

NOTES:

This cable works for both the Apple Personal Modem and ImageWriter II, so all 8 pins are connected even though not all are used for each device.

Figure 8.8c Macintosh Plus to Personal Modem/ImageWriter II Cable



NOTES:

Apple Computer Inc. does not connect the protective ground [PG] pins on their Macintosh to ImageWriter cable but we recommend it—it provides extra protection for your equipment.

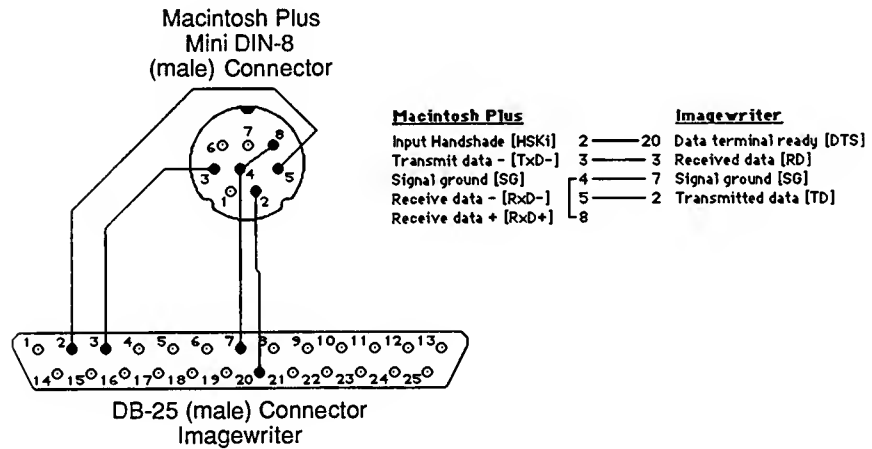
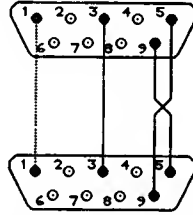


Figure 8.8d ImageWriter Cable

Macintosh
DB-9 (male) Connector



Macintosh
DB-9 (male) Connector

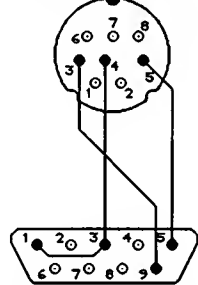
Macintosh

Protective ground [PG] 1
Signal ground [SG] 3
Transmit data - [TxD-] 5
Receive data - [RxD-] 9

Macintosh

Protective ground [PG] 1
Signal ground [SG] 3
Receive data - [RxD-] 9
Transmit data - [TxD-] 5

Macintosh Plus
Mini DIN-8
(male) Connector



Macintosh
DB-9 (male) Connector

Macintosh Plus

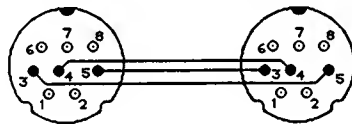
Transmit data - [TxD-] 3
Signal ground [SG] 4
Receive data - [RxD-] 5

Macintosh

Receive data - [RxD-] 9
Signal ground [SG] 3
Transmit data - [TxD-] 5
Protective ground [PG] 1

Mini DIN-8
(male) Connector

Mini DIN - 8
(male) connector



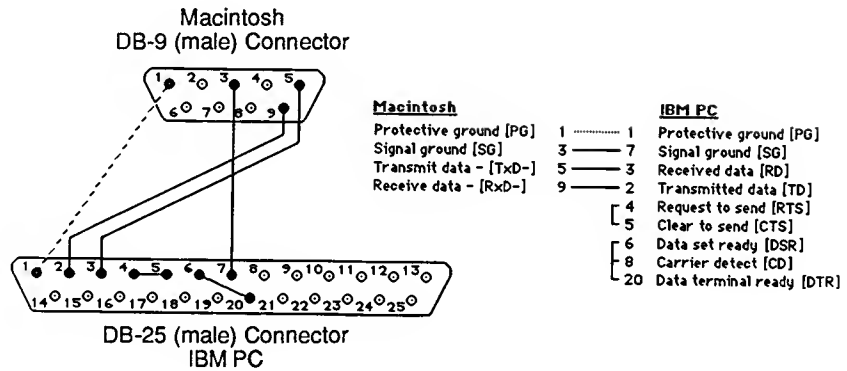
Macintosh Plus

Transmit data - [TxD-] 3
Signal ground [SG] 4
Receive data - [RxD-] 5

Macintosh Plus

Receive data - [RxD-] 9
Signal ground [SG] 4
Transmit data - [TxD-] 3

Figure 8.8e Direct Connect Cable



NOTES:

Dataviz Inc. does not wire a Protective ground [PG] in their cable, but it's a good idea to do so—it provides extra protection for your equipment. The jumpered pins at the IBM end of the cable (4-5 and 6-8-20) “trick” the IBM into thinking it is handshaking with another device.

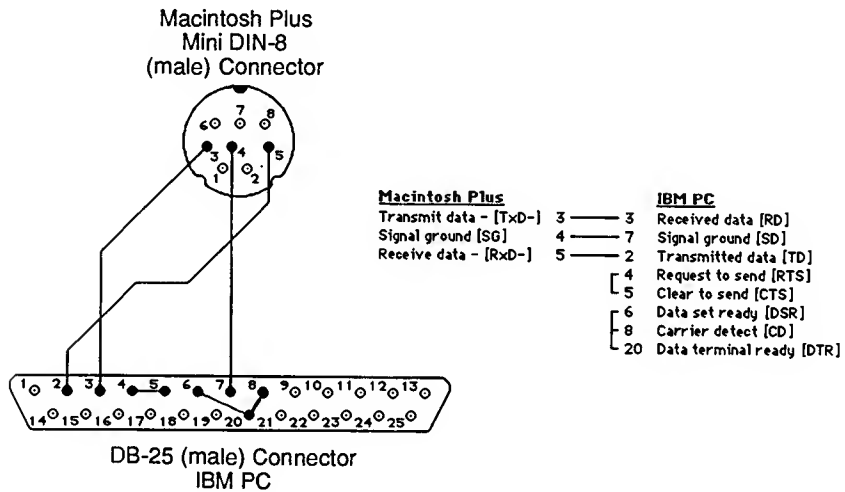
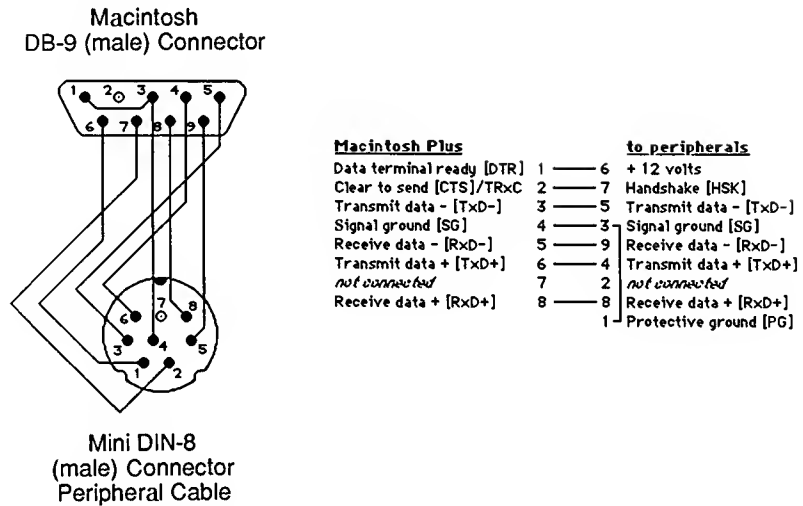


Figure 8.8f MacLink Cable

**NOTES:**

Ground pin 8 [RxD +] on the Macintosh Plus to emulate RS-232.

We will be offering this design as part of our line of high-tech jewelry—soon to be available in fine jewelry stores nationwide.

Figure 8.8g Macintosh Plus Adapter Cable

Diagnosing Cables

If you already have a cable that you are using to link either your modem and Macintosh, or your Macintosh and another computer, you may want to find out how it is wired so that you can either duplicate it or troubleshoot your connections.

There are three ways to do this. The first is to simply look in the manuals that accompany the devices you are linking. If the cable came with the device or works well with it, there may be a wiring diagram in the manual.

If the wiring diagram isn't in the manual or you don't have a manual to begin with, the second, least complicated way to determine the cable's wiring is to visually inspect it. By taking the casing off the connectors and looking at the wires, you can tell which wires are connected to which pins. Pin numbering at both ends follows a convention. Sometimes the pin numbers are embossed right on the connector. The wires are color-coded. For example, if the wire on pin 5 of the

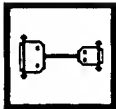
RS-232 connector is green, then look for the green wire at the 422 end and note the pin number. Do this for all the connected wires until you have reconstructed the wiring diagram.

The last resort, if you can't remove the housings at the ends of the cable, is to diagnose your cable using a voltmeter. By probing each pin in turn, you can not only tell which pin at one end is connected to which pin at the other, but you can also tell if a particular pin is cross-connected or *jumpered* to another. Inexpensive voltmeters are available at your local electronic supply store.

Using a Voltmeter If you are using a multipurpose meter, turn the dial to one of the ohm-reading settings (shown by the ohm symbol Ω). Plug the black wire to the COM outlet and the red wire to the ohm outlet (also marked by Ω). Connect the terminal (probe) end of the black wire to the first pin of one end of the cable.

Then, one-by-one, touch the terminal (probe) end of the red wire to each pin of the connector at the other end of your cable. When you reach a pin that's connected to the one attached to your black terminal, the meter pointer will indicate that current is able to flow through them and register on the voltmeter. You know that the two pins you're testing are connected.

Be sure to continue to test the rest of the pins with the red terminal. Some cables have two or more pins wired together at the same end (jumpered) inside the casing. When you've tested all the pins at the red end, move the black terminal to the second pin of that end of the cable and repeat the process. Make a sketch of the wiring as you go along.



The SmartCable A SmartCable is an easy way to connect two RS-232 devices. While you may not need to use it with your Macintosh, it comes in handy when you are trying to link up two new machines such as a modem and another micro. The SmartCable comes with its own circuitry and switches that make it a universal cable capable of linking just about every known RS-232 device.

It costs about as much as a custom-made cable, but if you find yourself swapping and testing cables a lot, it can pay for itself many times over. Certainly, if you have a professional interest in telecommunications and the practical problems thereof, you should have a SmartCable in your toolbox.

IQ Technologies (Appendix C) also makes a number of other intelligent cables and switchboxes for a variety of uses. Check them out.



Wiring Your Own Cables

If you find yourself becoming the *de facto* telecom expert on your block or in your work group, you'll be asked how to hook up different machines. Or if you find yourself working with many different kinds of machines over the course of your work, you'll want to be able to connect any two machines you come across.

Then, too, you may want to save money. It helps if you have a strong do-it-yourself streak in you. You'll be joining the ranks of luminary tinkerers such as Ben Franklin, Thomas Jefferson, or Steve Wozniak. When it comes right down to it, you may not be able to find a cable ready-made to your specifications, and you may not want to wait for someone else to make one for you.

Whatever the reasons, you *can* wire your own cables. Although there are other methods of assembling cables, the most common way to make one is to solder the appropriate connectors to each end of a cable cut to your needed length.

The cable diagrams we've provided show you how to wire some of the most common cables. Further, you can use our handy cable diagnostics outlined earlier to figure out how to duplicate an existing cable. (Preferably one that you have already tested in your situation so you know it works.)

Soldering is not particularly complicated, but it does require practice, patience, and concentration on detail. The tips here will help you get started if you've never soldered before. Practice on some spare wires and connectors before you attempt a finished cable.

- Be sure you have the right tools for the job.

25-watt soldering iron A 25-watt iron is hot enough to heat the parts sufficiently while not melting the plastic insulation (and other things you'd rather not melt) as quickly as higher-wattage irons.

Soldering iron holder When you're not actually using the soldering iron during a session, you need something that will hold the hot iron. These look like wire versions of a curled pig's tail, set on a heavy base so they won't fall off the table and into your lap, where the iron would melt things you'd rather not.

Damp sponge Use the damp sponge to wipe off the tip of the iron when it discolors during use. The tip of the iron should always be *tinned*. This means that you apply a thin coat of solder to the tip, which prevents the tip from oxidizing and provides good heat transfer to wires.

Rosin-core (electronic) solder Rosin-core solder is made specifically for use with electronic equipment. Acid-core solder should never be used. It is harmful to circuits and humans.

Helping hand A couple of small clamps on a heavy metal base helps hold the wires and connections in place as you solder. You can find these at most electronic parts stores.

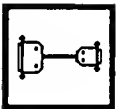
X-acto™ knife and wire strippers You will need tools to cut wires and to strip off the insulation properly from the ends before soldering them.

Small screwdrivers The connector housings have small screws that keep them assembled once soldering is complete. The housings also protect the connections inside.

Smooth file, fine sandpaper, or steel wool Before soldering a wire, it needs to be clean and free of oxide (discoloration due to exposure to oxygen in the air).

- Heat the wire, not the solder. Use the point of the soldering iron to transfer heat to the wire until the wire is hot enough to melt the solder. Never use the tip to melt the solder unless you are tinning the tip itself. When you apply the solder to the tip of the wire or to a connector, it should melt and flow evenly over the surfaces. This gives the best electrical connection.
- Cut and trim the wires evenly. Make a clean cut of the cable and strip about 1/4 inch of the plastic insulation from the end of each individual wire. Twist the wires together into tight, straight bundles.
- Heat and tin the wire tips individually. Once they are tinned, clip them so that only about 1/16" remains beyond the insulation.
- Heat and tin the cups of the pin connectors, filling each with solder.
- When making the actual connections, heat each cup of solder on the connector in turn and quickly plunge the right wire into the melted solder cup. Remove the heat and hold until the solder hardens. A nicely soldered connection should result.

If all that is just too intimidating and you *still* want to make your own cables, ask your local electronic supply shop about *solderless connectors*. You'll pay, but there's a lot less mess and bother.



Gender Changers

If you have an assortment of RS-232 to Macintosh cables, make sure you label each one and keep a record of how it's wired. If you get a reputation among your computer-using friends and associates as one of those whizzes who can make any file dance between any two machines, you'll find your cable collection to be quite a reputation enhancer.

Whenever you need to direct-connect a new machine (one you've never seen or used before) to your Macintosh, check to see if one of your existing cables will do the job before assuming that you have to build a new one. The most frequent problem is that you have a cable that's wired correctly for the two machines, but the connector plug at one or both ends of the cable is wrong for the machine(s) you are mating. The connectors are sometimes described as either "female" or "male" depending on whether the connector has holes or prongs that fit into holes.

There is a mass-produced solution to this problem of mismatched connectors, called a *gender-changer*. Gender-changers do not change the way a cable is wired. They just serve as an adapter bridge between the connector at the end of the cable and the connector on the computer. Gender changers come in three basic configurations: male on both sides, female on both sides, or a male/female combination. Try to have one of each around.



Synchronous and Asynchronous Modems

Until the mid 1980s, anyone who had to move large amounts of data between two remote locations had to use *synchronous* modems on conditioned, dedicated telephone lines. Synchronous modems use a timing signal to get both sides of the transmission in sync. The addition of the timing signal requires more complex circuitry in the modem, but it eliminates the start and stop bits that are necessary in asynch modems. In addition, asynch modems must exchange acknowledgment signals during transmission, further degrading their overall efficiency. Synchronous modems, then, are primarily used between mainframes. They allow the exchange of information at rates up to 9600 bps. As stated earlier, to get these transmission rates, companies usually have to lease a special line from the phone company that is less noisy and more reliable than an ordinary voice-grade circuit.

Another advantage to using synchronous modems is that information exchange can flow in both directions at once, taking full advantage of the duplex channel.

However, the distinctions between synchronous and asynchronous modems are fast eroding as technology proceeds at its usual breakneck pace. It is now possible to get the same speed advantages using asynch modems that were once only possible using expensive dedicated lines and synchronous transmission. Furthermore, bidirectional transmission is also available using asynch communications with

packages such as BLAST, which uses a synchronous protocol in asynch mode. Using the BLAST protocol with even a 1200 bps asynch modem achieves results that were previously only available with synchronous methods.

Telebit's TrailBlazer Modem

Until recently, the fastest transmission speed a telecom user could expect using existing technology and ordinary voice-grade telephone wires was 2400 bps. The technical experts stated flatly that higher speeds on such unconditioned, noisy channels were technically impossible.

Even with today's 2400 bps modems, effective transmission rates are often much lower than 2400 bps owing to the fact that if the lines are exceptionally "dirty," information will have to be retransmitted until it gets through without errors.

The Telebit Corporation has broken the voice-grade telephone line speed barrier with its TrailBlazer and TrailBlazerPC modem series. (The latter is a plug-in board for use with IBM PCs and compatibles.) Telebit's technology allows speeds of 10,000 bps (10K bps) and *higher*.

As this is being written, the only terminal software packages on the Macintosh that support Telebit's breakthrough are MicroPhone and inTalk. To achieve these fast transmission rates, of course, both sides must be using the Telebit modem and software that is also capable of supporting these faster speeds.

As shown in Figure 8.9, Telebit achieves these high data transmission rates by using more of the available bandwidth than other modems do. In use, the TrailBlazer modem first analyzes the line conditions across the spectrum of available frequencies. Based on what frequencies are useable, the modem determines which of its modulation schemes it will use. It then assembles *packets* of bits. (A packet is any group of bits used in a transmission protocol. A given protocol such as Xmodem, for example, will group bits in packets of 128 bits, 1K bits, and so on.) These packets are then sent on their way using a proprietary error-checking scheme employing Cyclic Redundancy Checking (CRC).

Right now, this technology is more expensive than 2400 bps modems. Suggested list price for a TrailBlazer modem that works with the Macintosh is \$2,395. The TrailBlazer also supports the standard speeds of 300 and 1200 bps, making it compatible with other modems. We expect, however, that the cost curve of Telebit's technology will follow the familiar downward slope that has historically marked electronic products.

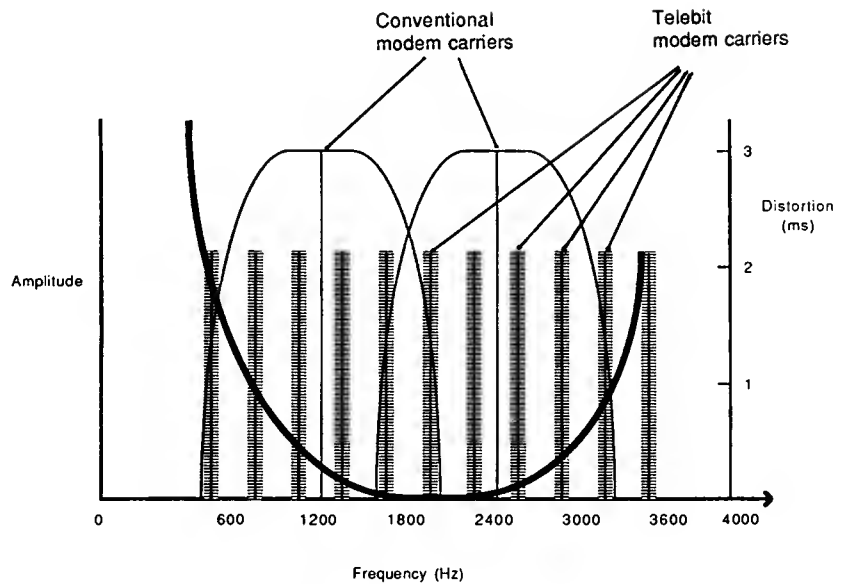


Figure 8.9 Telebit Bandwidth



Error-Checking Transfer Protocols

Xmodem and Ymodem

Xmodem was an outgrowth of a program called MODEM.COM, written by Ward Christensen to run on micros using CP/M. It was a short terminal program designed to handle sending and receiving files. While it incorporated an error-checking protocol which later became Xmodem, it was never intended for use outside of micro-to-micro exchanges. Christensen himself stated, "I wasn't out to invent something famous, just to hack something together so I could transfer some files." About 1977, he threw it into the public domain and then walked away. What has since become known as the Xmodem protocol was born at the hands of Keith Petersen of Royal Oak, Michigan. Petersen took Christensen's MODEM.COM program and stripped it down to "send" and "receive." Since then, Xmodem has been implemented on a wide variety of micros and mainframes, including services such as CompuServe and the Source. Christensen's name continues to be associated with the protocol to this day, even though Petersen did the hacking that made it what it is. It should rightly be called the Christensen/Petersen Protocol.

Xmodem's structure and approach to data transfer is similar to that taken by most error-checking protocols. Xmodem is packet-oriented. That is, data is sent in packets or blocks of 128 bytes each. (Why 128? Because the CP/M disks, with which the protocol was originally designed to work, had sectors of 128 bytes each.) Each packet is numbered as it is assembled by the sending computer. A checksum is appended at the end of the packet. At the receiving end, the target micro performs its own checksum on incoming blocks and compares its result with the sender's. If the checksums do not match, the receiver requests that the block be resent. This process continues until the block is received without errors.

When the block is received error-free, the target computer will send the transmitting computer an *acknowledged* signal: ACK (ASCII 6 character, ^F).

At the beginning of the exchange, the receiving computer begins a time-out sequence and sends a character signifying that it is ready to begin getting blocks of data. That is, it begins watching its internal clock and sends a NAK (ASCII 21 hex, ^U), then waits for the initializing signal from the transmitter, in this case an SOH (ASCII ^A, for *start of header*). It continues to send a NAK every 10 seconds until the sender begins transmitting a block. If the sending computer doesn't send an SOH character within a set period of time, the receiver quits waiting and the process has to be restarted.

The checksum is calculated by adding the numbers for the SOH character, the block number, the block number's two's complement, and all 128 bytes of data.

If the sending computer sends a block and the receiver gets it and sends an ACK, but for some reason the ACK doesn't reach the sender, then the sending computer will think it has failed and will resend the block. Now the receiver is one block ahead of the transmitter. They are said to be out of synch. In this case, the receiver throws away the extra block and resends an ACK.

Transmission ends when the transmitter sends an EOT character (end of transmission, ASCII 4).

Under most conditions where the line noise is at an acceptable level and the receiving computer can keep up with the transmissions, Xmodem works quite well. However, Xmodem has several drawbacks under certain circumstances.

If the receiving computer is busy and can't keep up with the sender for any reason (it's dealing with a large number of users, for example), or if the transmission distances are very large making the

transmission times correspondingly long, or if the lines are exceptionally noisy, Xmodem breaks down. The sending computer may time out before the receiver has a chance to acknowledge a block, and the whole process will have to begin all over again. In fact, if transmission is interrupted for any reason before the whole file has been sent, Xmodem requires that the transmission be restarted from scratch.

To get around these problems, some implementations of Xmodem have been modified slightly. Relaxed Xmodem, for example, allows the time-out periods to be lengthened (the normal time-out is 1 second) to accommodate exchanges with busy systems. Some implementations also allow the transmission of blocks longer than 128 bytes (usually 1K bytes instead) in order to cut down on the number of ACKs, NAKs, and checksums that must be performed. This also speeds up Xmodem somewhat.

Another drawback to Xmodem is that, in some situations, its checksum algorithm may allow errors to creep into what is supposed to be an error-free transmission. To augment its checksum procedure, some implementations of Xmodem include additional error checking called a cyclic Redundancy Check (CRC) to each block of data sent.

A CRC version of Xmodem will send out C's at the beginning of transmission instead of NAKs. If the sender has a CRC option available, that option treats the C like a NAK, but knows that it has to type the CRC and add it to the end of all the transmitted blocks. If the sender isn't able to handle the CRC option, then the C character will be meaningless to it and it will be ignored. The receiver then knows that the sender doesn't have CRC and will then switch to the normal NAK/ACK sequence to start the transfer.



CRC Cyclic redundancy checking is a mathematically more powerful error-detecting and error-correcting method than that used by Xmodem alone. With CRC, as with simpler checksumming methods, a character is generated based on a computation done on the contents of the message. At the receiving end, a similar mathematical manipulation is performed, and if the numbers match, the message is probably correct. This method of error checking can detect all odd numbers of error bits, all possible single-error bursts not exceeding 16 bits, 99.9969% of all possible single bursts 17 bits long, and 99.9984% of all possible longer bursts. (It's still not perfect.) To find out more about the

precise calculations performed, see "Cyclic Redundancy Check" in the *Encyclopedia of Computer Science and Engineering* (p. 434).

To get around the problems inherent in Xmodem, which was designed for micro-to-micro transfers over phone lines and not for micro-to-mainframe exchanges over packet-switched networks such as Tymnet, other protocols have been developed, including Kermit, BLAST, and X.PC. Xmodem is not very good when it comes to using high-speed modems, either. At transmission rates above 1200 bps, it effectively slows down the exchange considerably. Xmodem is a half-duplex protocol: information flows in one direction at a time. X.PC and BLAST (and Yves Lempereur's Multichannel Communication System) do not suffer from this drawback.

The Ymodem enhancement of Xmodem allows multiple files to be exchanged without the need to restart and respecify the transmission parameters between each file. Ymodem has been implemented on CompuServe. It is not widespread. We don't know of any Macintosh terminal program that supports it other than MicroPhone.



Kermit

Kermit is a protocol for transferring files between computers of all kinds and sizes using asynchronous modems and ordinary telephone lines. It specifies packet sizes (the number of bits to be sent in each block of bits; the terms *packet* and *block* are used interchangeably), checksums (a calculation that is used to check for errors), and data retransmission (when an error is found in a packet, the packet is resent).

Kermit is nonproprietary, widespread, and thoroughly documented. It is more robust than Xmodem when used with high-speed modems and mainframes. It was originally developed at Columbia University, which still acts as a clearinghouse for information about Kermit and new Kermit implementations.

Kermit software is free. However, you must pay a fee to get Kermit from Columbia. The fee covers costs associated with printing documentation, distribution media (tapes, paper, etc.), postage, and handling.

The user's guide contains complete instructions for installing Kermit. The *Protocol Manual* is a guide for writing a new implementation of Kermit and incorporating it into your own software product.

Kermit is available on nine-track magnetic tape in any of the following formats (as of this writing):

System	Tape Format	Density
VAX/VMS	ANSI Label, Format D (VMS COPY)	1600
UNIX	TAR	1600
TOPS-10	BACKUP/Interchange, Unlabeled	1600, 6250
TOPS-20	DUMPER, Unlabeled	1600, 6250
IBM VM, MVS	EBCDIC, OS Standard Label	1600, 6250
IBM VM/CMS	EBCDIC, CMS Tape Dump Format	1600, 6250
All Others	ASCII, ANSI Label, Format D	1600

When you order, you must specify the system, format, and density you require for your mainframe. Columbia cannot supply formats other than those listed. Fixed record formats, 800 bpi, or unlabeled tapes are not available. They cannot produce floppy disks, either. They provide bootstrapping procedures for creating microcomputer versions from the mainframe for which the tape is produced.

To order or to get more information, write:

Kermit Distribution Center
Columbia University Center for Computing Activities
7th Floor, Watson Laboratory
612 West 115th Street
New York, NY 10025

MS DOS-based (IBM PCs and compatibles) Kermit can be ordered from PC SIG (408-730-9291).

Note: Educators and university users please note that Kermit is also available to users of the BITNET education network at CUVMA (BITNET users type SMSG RSCS MSG CUVMA KERMSRV HELP for further information); on ARPANET (via anonymous FTP from host CU20B, in the area KER:); UUCP from host okstate 10pm-10am CST; and on magnetic tape from user groups such as DECUS and SHARE.



X.PC

To begin with, X.PC is a full-duplex protocol. File exchanges can flow in both directions at once. Its error-checking employs CRC. X.PC is also capable of synchronous communication, which checks the arrival of

data by using a timing signal rather than start and stop bits. This eliminates the two extra bits needed in asynch communication. X.PC can also be used with asynch exchanges.

X.PC was developed to conform to an international data exchange protocol called X.25. X.25 is a synchronous protocol. X.PC and X.25 will work together, however. X.PC also has the advantage of being able to support multiple communications exchanges over a single transmission channel. This is called *virtual circuiting*. For example, using a network that supports X.PC, you could have one window on your Macintosh showing an exchange with Dow Jones News Retrieval, getting your current stock data, and a second window providing an exchange with one of your clients. Packets of data would be automatically routed to the correct location and the correct window—and all this on the same telephone line!

Currently, only Tymnet has implemented X.PC but with only one virtual circuit (as of this writing). Support for X.PC has been announced by MCI Mail, Microsoft, and Microstuf, among others. Tymnet has placed X.PC in the public domain, which means that anyone can implement the protocol without licensing or use fees, a factor that will most likely contribute to widespread use.



MacBinary

Source: Dennis F. Brothers, Rev. 3, 6 May 1985

During April of 1985, an informal working group of Macintosh developers with interests and expertise in telecom was formed to discuss and refine Dennis Brothers' proposal for a new MacBinary format for information exchange.

The resulting MacBinary standard format is intended for use between Macintoshes running terminal programs (possibly different ones) which adhere to the standard. It is also for use in exchanging arbitrary Macintosh documents with remote systems where it is presumed that the transmitted documents will be stored as an exact image of the data transmitted. The standard also provides for processing to be performed on text files transferred via a protocol, in order to maximize the likelihood that text files sent to a remote system will be useable by that system, and that text files received from a remote system will be useable by the Macintosh.

MacBinary has become a standard. By permitting direct transfer of Macintosh documents without any intermediate need for file consolidation or conversion, it cuts down the amount of time and effort

required to share information between Macintoshes. MacBinary file transfers take 25 percent to 50 percent less time to move, too.

MacBinary is independent of the communication protocol used to accomplish a transfer. It assumes only that an 8-bit transparent transfer can be accomplished (see Chapter Four). Such protocols as Xmodem (Christensen/Petersen), Kermit, and CompuServe A or B meet this requirement.

The MacBinary format (Table 8.1) consists of a 128-byte header containing all the information necessary to reproduce the document's directory entry on the receiving Macintosh; followed by the document's data fork (if it has one), padded with nulls to a multiple of 128 bytes (if necessary); followed by the document's resource fork (again, padded if necessary). The lengths of these forks (either or both of which may be zero) are contained in the header.

The format of the 128-byte header is shown in Figure 8.10.

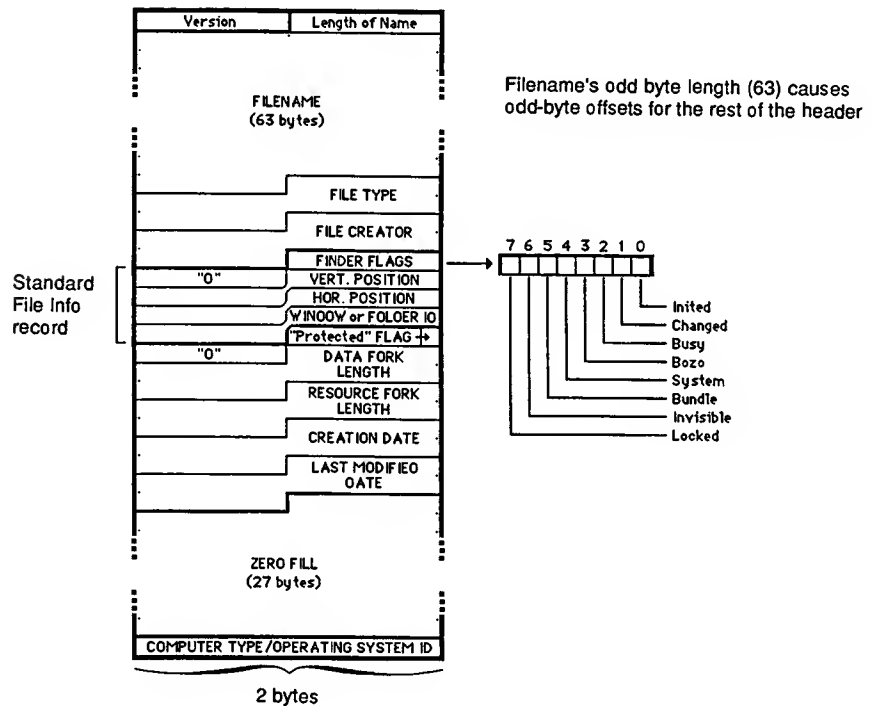


Figure 8.10 MacBinary Header

Table 8.1 MacBinary Format

Offset (decimal)	Length (decimal)	Contents
000	1	Zero. this is a "version" byte
001	1	Length of filename
002	63	Filename (only "length" bytes are significant)
065	4	File type
069	4	File creator
073	1	Finder flags: Bit 7—Locked Bit 6—Invisible Bit 5—Bundle Bit 4—System Bit 3—Bozo Bit 2—Busy Bit 1—Changed Bit 0—Initd
074	1	Zero
075	2	File's vertical position within its window
077	2	File's horizontal position within its window
079	2	File's window or folder ID
081	1	"Protected" flag (in low-order bit)
082	1	Zero
083	4	Data Fork length (bytes, 0 if no Data Fork)
087	4	Resource Fork length (bytes, 0 if no Resource Fork)
091	4	File's "creation" date
095	4	File's "last modified" date
099	27	Zero fill (reserved for expansion of standard)
126	2	Reserved for computer type and OS ID (0 for the current Macintosh, 1 for the Mac Plus)

It is the responsibility of the receiving terminal's program to resolve file name conflicts, generally by modifying the name of the received file if there already exists a file with the original name on the target volume. The most common practice is to rename it by adding a *#n* or via *<software name>*.

Note that while the original window or folder ID and position may be transmitted, the receiving terminal program would not normally set these items for the received file. It would instead accept the values that the Macintosh File Manager assigns when it creates the new file.

When receiving a file via the MacBinary protocol, a terminal program may distinguish between text and document modes by examining bytes 0, 74, and 82 of the first 128 bytes received. If they are all 0 (and at least 128 bytes have been received), then it is safe to assume that a MacBinary formatted document is being received.



Multichannel Communication System

The Multichannel Communication System (MCS) was developed to handle file transfers over satellite links with their high noise levels, long transmission delay times, and high costs. MCS was developed by Yves Lempereur (who, you may remember from our previous discussion, also developed BinHex 5.0) for use by Mainstay to communicate between California and France. It is similar to the X.25 protocol developed for international data communication. It is also similar to BLAST in that it allows two-way, simultaneous, error-free transmission of files. Unlike BLAST, however, while files are being swapped, people at both ends can “chat” at the same time without affecting the file exchange transmission rate. In other words, MCS creates three *virtual channels* out of one physical link: one to chat, one to send, and one to receive.

MCS uses MacBinary format for transfers. It is a reliable file transfer method for use by those who might want to experiment with the possibilities inherent in two-way simultaneous data telecom or who have overseas communication needs similar to Mainstay’s. It, too, is a shareware product at the time of this writing. For information on cost and how to obtain a working copy of MCS, contact:

Yves Lempereur
28611B Canwood Street
Agoura Hills, CA 91301
Or call (818) 991-6540



Data Security and Encryption

No one is immune from computer-related breaches of privacy. If you are careless with your information service account identification numbers and passwords, you can be “ripped off” and never know it. When most people think of computer crime, they think of teenagers breaking into a mainframe belonging to the government or private industry. (Attribute this to mass media “hacker hype.”) However, as the value of information becomes more widely appreciated, individuals who use mainframe services and the phone lines to conduct their business will become increasingly vulnerable.

If you deal with proprietary information or private information that must, by law, be protected, you must begin to consider information security now, before you get caught in an unpleasant scenario. If

you maintain large quantities of information in your private data base, information that can be considered sensitive and private to other individuals, then you are at risk. The persons about whom your information is kept are also at risk and may hold you liable if information about them is misused and/or causes them financial damage.

When you begin shipping such information over the phone lines, you are increasing the risk. As this is being written, Congress is considering updating the Omnibus Crime Control Act to bring electronic data communication under the protection of the Constitution and our First Amendment. The Justice Department is resisting this, and at this time, what the eventual outcome will be remains unclear. In the meantime, data communications can be easily tapped by government agencies for any reason. There is nothing to stop individuals within those agencies from misusing *your* private information.

What measures can be taken? Quite a few. In addition, users of telecom technologies can also begin to press for additional protective measures that do not depend on the largess of Congress for their efficacy.

First of all, protect your passwords, identification numbers, and associated materials zealously. If you have embedded your passwords into programs that are stored on disks from which they can be used to log onto another system automatically, then you should maintain the physical security of those disks just as you guard access to your credit cards and checkbooks.

If you have stored financial information about other people on disks (say you are a stockbroker who keeps client portfolios on your Macintosh Dow Jones program disks), then, likewise, you should take steps to secure access to those physical artifacts.

If you use an electronic mail system, whether privately run (by your employer for in-house communications, for example) or publicly accessible (such as MCI Mail or Source Mail), you should know that your files are not secure from prying by individuals within these organizations or, necessarily, by other users of the systems. To protect yourself, consider encrypting all sensitive text files you transmit or store in such systems.

The most secure method is to use a separate password key for each file and each individual with whom you exchange data. The passwords should be transmitted to the other person through a channel known to be secure, or reasonably so. For example, let's say you want to send electronic mail to a friend through MCI Mail. First, create a list of passwords consisting of ten characters each. The list can be as

long as you want but, in practice, twenty to fifty are convenient to work with. Next, send the list to your colleague through regular mail channels. Now when you want to send a letter or other kind of data file, you first encrypt it (with one of the readily available encryption programs on the market) using the first password on your list. The encryption process turns the file into gibberish that can't be read until it's decrypted using the same password. Your colleague picks up the file and decrypts it, again using the first password on the copy of the list you've provided. Then you both discard the first password and never use it a second time. When your associate sends back a reply, he or she first encrypts it, this time using the second password on the list, and so on.

This method is highly reliable for most ordinary purposes. In any event, if you are subjected to snooping by an agency that has the time, money, and power (a supercomputer running part-time), even these methods won't suffice.

There are special modems available that will encrypt information on the way out of your system. These must be used in conjunction with identical modems running at the other end. This solution is quite expensive and justifiable only if you are in a business or industry that requires these measures. For most individuals and businesses, using the one-time password method described above will secure the data.



N'Cryptor

An effective way to secure your Macintosh files, whether they are intended for exchange over phone lines or will remain resident with your Macintosh, is to use N'Cryptor from Mainstay. Even if the files are stolen or intercepted, a would-be thief will be unable to use them without the encrypting password. For current price and availability contact:

Mainstay
286111B Canwood Street
Agoura Hills, CA 91301

Check your local computer user group, too, for other encryption programs that may be available.

Overall security provisions can help protect your information from most wrong-doers. Here's a checklist of precautions you may want to take.

- ✓ **Access Security:** prevents unauthorized access to the machines, disks, and documentation. A locked room or locked cabinet is an example.
- ✓ **Communication Security:** provides protection from electronic eavesdropping. Employing modems that scramble data transmissions or employing the one-time pad method fall into this category.
- ✓ **Data Backup:** provides protection from accidental or deliberate erasure of sensitive or critical information. Backup your important disks and store them off site. Fire, floods, earthquakes, and hackers can then do their worst without corrupting or destroying your information.
- ✓ **User Identification:** ID codes and passwords that verify a user's right to access your system. This is the most widely used and most easily breached form of protection. But just as you wouldn't consider going on vacation without locking your front door, don't overlook this elementary form of security.
- ✓ **User Location Correlation:** sometimes called a *ringback* or *callback* system. The user's telephone number is stored along with an ID number and password. When the user calls in and validates his or her user information, the system hangs up and redials, using the stored phone number.
- ✓ **Device Pairing:** two modems, using synchronized encryption techniques, that can only be used together. Strange modems get gibberish on the line. The same net result can be accomplished by using the one-time password technique. As far as we know, there is no terminal software that will accomplish the same thing.



Standards Organizations

There are many standards published by many groups that affect the telecom world. Some of those standards, as we have seen, actively compete with one another, which is probably to the good as far as the evolution of technology is concerned. After all, too early a standard can retard further progress. A standard adopted too late can prevent the spread of a useful technology. Ultimately, it is the world marketplace, with its thousands of individual choices, that determine what standards spread and which are aborted.

In the telecom world, the number of national and international standards bodies is quite small. Here are the ones you may encounter most frequently.

ANSI The American National Standards Institute is an organization that develops and publishes industrial standards. In telecom, ANSI's major objective is defining the characteristics of digital data communication systems including those of data transmission. U.S. contributions to the ISO (below) are made through ANSI.

CCITT The International Consultative Committee on Telegraphy and Telephony. This is an international organization that formulates and proposes recommendations for international telecom standards. CCITT also monitors international telecommunication, including the compatibility of data communication and data processing systems.

EIA The Electronic Industries Association. This group is made up of representatives from major commercial companies in the electronics industries. It formulates technical standards, disseminates marketing data, and represents industry to government.

ISO The International Standards Organization is a group made up of representatives of each of the national standards bodies of 95 countries worldwide. Only one organization per country may belong to this group. Participating members work on committees and can vote for or against proposed standards. The U.S. representative organization to the ISO is ANSI.

In addition to these groups, AT&T wields a lot of clout in the forming of standards. The two common modem standards, Bell 103 (110 bps modems) and Bell 212 (1200 bps modems) were set by Bell and are followed by just about every U.S. manufacturer of modems today.

The federal government also influences standards through the military and the National Bureau of Standards. The latter publishes its standards under the acronym FIPS, which stands for Federal Information Processing Standards. Most FIPS are developed in cooperation and are compatible with industry standards.

For a synopsis of standards and standards bodies, contact:

Case Rixon Communications
2120 Industrial Parkway
Silver Spring, MD 20904

This company publishes *The Data Communications Standards Handbook*.

Epilog: TeleFutures



After the wires grew all over the ground, they began to interconnect. About the year 1980, the interconnections became about as complex as the nervous system of a flatworm as measured by the number and variety of connections. Meantime, sensor and effector technology became more sophisticated.

Electronic machines could read. They could speak. They could sense temperature, humidity, atmospheric pressure, radiation, a wide range of chemicals (i.e., they could smell), brainwaves, radio waves, and every kind of humanly encoded electrical signal.

Between 1940 and 1990, the variety of “end-result” boxes that human beings interacted with reached its peak, and began to reintegrate and shrink in size at a rapid rate. At the height of this era, the number of separate, “remote-control” boxes that could be found in the average consumer’s home reached a mean of 13.6 per household, or 4.4 per person. Therapists reported a rise in “remote confusion syndrome” stemming from the anxiety inherent in trying to remember which devices and functions were used with which hand-held remote control. Up until then, what were separate instruments (telephones, radios, televisions, videotape recorders, personal computers, cameras, answering machines, cellular phones, satellite downlinks, and personal entertainment devices such as the portable cassette and compact disc player) began to show up in integrated “boxes.” Each line of boxes, for a

time, contained different combinations of media and sensors. For example, one line of development integrated still and motion picture cameras into high-definition video imaging boxes. The so-called integrated work station of the mid 1980s became the desktop publisher of the late 1980s and early 1990s. The desktop publisher combined with the personal messenger which in turn became the cellular processor, and then, at long last, today's Life Processor.

Contemporary scholars look upon the proliferation and recombination period from the mid 1980s to the late 1990s to be very rich source material for discourse analysis and media ecology studies. The paleomedia era (pre-electromagnetic spectrum days, writing/print and earlier) had been pretty much worked over in terms of thesis possibilities by the early twentieth century. However, the mesomedia era, which is this span between Maxwell's equations and the Life Processor referred to earlier, remains an inexhaustible source of correlatable material. As government documents from that early time continue to become declassified and enter the public domain, the picture of the relationships between media, politics, economics, and contemporary global cultural phenomena becomes more detailed.

It is no coincidence, for example, that the rise of the white-collar growth cults took place during this era of proliferating personal media. Selling a combination of communications skills and cybernetic information, these cults rode the wave of audio and video cassette popularity. In one bizarre case, a middle-eastern fundamentalist religious figure used ordinary audio cassettes to topple a state.

But that, as they always say, is another chapter in some other book

Integrated Life Processor Function List

Telephone

Biofeedback and health monitoring

Calorie and exercise monitoring

Language translation

Microcomputer

Communications (Total Integrated Link Translator (TILT))

Video and audio, video only (VO) and audio only (AO) modes

Reading (Kurzweil Circuitry) scanner

Cellular telephone with access to ISDN (Integrated Services Digital Network)

Smart Credit Card

Mega memory

Data bank and e-mail access

Pager/beeper

Phone secretary (phone answering and voice synthesized response)

Encryption and failsafe mechanisms

Optoelectronic Dynabooks with writable laser disk storage

Instant access to human networks of

counselors

therapists

teachers

trainers

career and investment planners

calendar

Guardian Angel circuitry consisting of

police; fire (smoke detector); home monitor/remote

ambulance (linked with biomonitoring circuitry)

links to medical centers if needed

Links to external displays (large screen HDTV, amplifiers)

Links to external input devices (keyboards, cameras, data storage)

Links to appliances and power lines

Automobile link, with optimal routing and map planners

Bank account links

The Life Processor is keyed to thumbprint/voiceprint combination, plus appropriate encryption and self-destruct passwords. These devices are constitutionally treated as part of the definition of an individual, hence they are protected on "cannot testify against oneself" grounds. No law enforcement official may use the contents of a Life Processor in any legal proceeding. Moreover, mere possession of any other individual's registered Life Processor is a felony punishable by 10-year jail terms.

For more information about current and future developments in Life Processor machinery, see *Engines of Creation: Challenges and Choices of the Last Technological Revolution* (Drexler 1986).

APPENDIX



FILE to FILE Import/Export Charts

Product **Art Grabber with
Body Shop**

Company *Hayden*

Category graphics utility

Data Types bitmap

Import bitmap

Export bitmap¹

Notes ¹Via Clipboard

Product **ClickArt Special
Effects**

Company *T/Maker Graphics*

Category graphics utility

Data Types bitmap

Import bitmap²

Export bitmap²

Notes ²Directly from within
MacPaint

Product **ClickOn Worksheet, TYPE1**
 Version: 1.3
Company *T/Maker Graphics*
Category spreadsheet
Data Types text, number
Import
Export
Notes

CLWD

CLW1

CREATOR1

Product **Crunch**
Company *Paladin*
Category spreadsheet
Data Types text, numbers
Import DIF, SYLK, WKS
Export
Notes

TYPE1

CALC

SOFD

CREATOR1

Product **Easy 3-D, Version: 1.00**
Company *Enabling Technology, Inc.*
Category graphics
Data Types bitmap
Import bitmap¹
Export bitmap¹
Notes ¹Via Clipboard

TYPE1

EZ3D

EZ3C

CREATOR1

Product **Ensemble, Version: 1.0**
Company *Hayden*
Category integrated
Data Types text, number, bitmap
Import text¹, bitmap¹
Export text¹, bitmap¹
Notes ¹Via Clipboard

TYPE1

MODL

CXCC

CREATOR1

Product **Excel**, Version: 1.00
Company *Microsoft*
Category integrated
Data Types text, number
Import text³, SYLK, WKS
Export bitmap¹, text, SYLK, WKS
Notes ³Can be delimited by nulls,
commas, returns, tabs,
spaces
¹Via Clipboard

TYPE1	TYPE2	TYPE3
XLBN	TEXT	TEXT
XCEL	XCEL	XCEL
CREATOR1 regular	CREATOR2 SYLK	CREATOR3 text

TYPE4
TEXT
XCEL
CREATOR4 WKS

Product **Factfinder**, Version: 1.1
Company *Forethought, Inc.*
Category database
Data Types text
Import text
Export text
Notes

TYPE1
FACT
NARU
CREATOR1

Product **File**, Version: 1.02
Company *MicroSoft*
Category database
Data Types text, number, date, bitmap
Import text⁴, bitmap¹
Export text⁴, bitmap¹
Notes ⁴Can be delimited by
returns, tabs
¹Via Clipboard

TYPE1	TYPE2
ISAM	KEYS
FILE	FILE
CREATOR1 data	CREATOR2 index

Product **FileMaker**, Version: 1.0 **TYPE1**
Company *Forethought, Inc.* **NUTD**
Category database **NUTS**
Data Types text, number, date, bitmap⁵
Import text⁶, SYLK, bitmap⁵ **CREATOR1**
Export text⁶, SYLK
Notes ⁵For decoration only; via Clipboard
⁶Can be delimited by commas, returns, tabs

Product **FileVision**, Version: **TYPE1**
Business **PicB**
Company *Telos*
Category database **TSP2**
Data Types text, number, bitmap **CREATOR1**
Import bitmap¹, text⁷, DIF, SDF, SYLK
Export bitmap¹, text⁷, DIF, SDF, SYLK
Notes ¹Via Clipboard
⁷Delimiters are user-definable

Product **Helix**, Version: Double **TYPE1**
Company *Odesta Corporation* **HEAP**
Category database **HELX**
Data Types text, number, date, flag **CREATOR1**
Import text⁴, SYLK, DIF
Export text⁴, SYLK, DIF
Notes ⁴Can be delimited by returns, tabs

Product Home Accountant , Version: 1.03	TYPE1 HSYS	TYPE2 CATG
Company <i>Arrays, Inc./Continental Software</i>	HAMC	HAMC
Category accounting	CREATOR1	CREATOR2
Data Types text, number	system	data
Import		
Export		
Notes		

Product **InterLace**, Version: 1.1
Company *Singular Software*
Category integrated database
Data Types text, number, date, time, boolean, bitmap⁵
Import text⁴, mailmerge⁸, bitmap⁵
Export text⁴, mailmerge⁸
Notes ⁵For decoration only; via Clipboard
⁴Can be delimited by returns, tabs
⁸Same as text but with mailmerge header info

TYPE1	TYPE2	TYPE3
SSRF	SSDB	TEXT
SING	SING	
CREATOR1 linked set	CREATOR2 table	CREATOR3 export

Product **Jazz**, Version: 1
Company *Lotus*
Category integrated
Data Types text, number, bitmap
Import text³, bitmap¹, SYLK, WKS
Export text³, bitmap¹
Notes ³Can be delimited by nulls, commas, returns, tabs, spaces
¹Via Clipboard

TYPE1	TYPE2	TYPE3
JTRM	JDBS	JFRM
JAZZ	JAZZ	JAZZ
CREATOR1 comm	CREATOR2 database	CREATOR3 form
TYPE4	TYPE5	TYPE6
JGRF	JWKS	JWPD
JAZZ	JAZZ	JAZZ
CREATOR4 graphic	CREATOR5 spreadsheet	CREATOR6 w/p

Product **MacDraft**, Version: 1.1
Company *IDD, Inc.*
Category graphics
Data Types text, bitmap, quickdraw
Import text¹, bitmap¹, quickdraw¹
Export text¹, bitmap¹, quickdraw¹
Notes ¹Via Clipboard

TYPE1
DRWG
MACD
CREATOR1

Product **MacDraw**, Version: 1.7 **TYPE1**
Company *Apple Computer, Inc.* **DRWG**
Category *graphics*
Data Types *text, bitmap, quickdraw* **MDRW**
Import *text¹, bitmap¹, quickdraw¹* **CREATOR1**
Export *text¹, bitmap¹, quickdraw¹*
Notes ¹Via Clipboard

Product **Mach v1.0**, Version: 1.0 **TYPE1**
Company *Palo Alto Shipping Company* **TEXT**
Category *programming language*
Data Types *user definable* **EDIT**
Import *text* **CREATOR1**
Export *text*
Notes

Product **MacPaint**, Version: 1.5 **TYPE1**
Company *Apple Computer, Inc.* **PNTG**
Category *graphics*
Data Types *bitmap* **MPNT**
Import *bitmap* **CREATOR1**
Export *bitmap*
Notes

Product **MacProject**, Version: **TYPE1**
1.0 **MPRD**
Company *Apple Computer, Inc.* **MPRJ**
Category *project management* **CREATOR1**
Data Types *text, bitmap⁵*
Import *text, bitmap⁵*
Export *text⁴, bitmap⁵*
Notes ⁵For decoration only, via
Clipboard
⁴Delimited by returns, tabs

Product	MacWrite , Version: 4.5	TYPE1	TYPE2
Company	<i>Apple Computer, Inc.</i>	WORD	TEXT
Category	word processing		
Data Types	text, bitmap	MACA	MACA
Import	text, bitmap ¹ , SYLK	CREATOR1	CREATOR2
Export	text, bitmap ¹	formatted	text only
Notes	¹ Via Clipboard		
Product	Magic Video Digitizer , Version: 1.2	TYPE1	TYPE2
Company	<i>New Image Technology</i>	PNTG	PATB
Category	video digitizer		
Data Types	bitmap	MPNT	MAGI
Import		CREATOR1	CREATOR2
Export	bitmap	MacPaint	pattern
Notes			
Product	Multiplan , Version: 1.1	TYPE1	TYPE2
Company	<i>Microsoft</i>	MPBN	TEXT
Category	spreadsheet		
Data Types	text, number	PLAN	PLAN
Import	text, SYLK	CREATOR1	CREATOR2
Export	text, SYLK	regular	SYLK
Notes			
Product	Omnis3 , Version: 3.10	TYPE1	TYPE2
Company	<i>Blyth Software</i>	OM\$D	OM\$L
Category	relational database		
Data Types	text, number, boolean, date	OM\$\$	OM\$\$
Import	text ⁶ , DIF, SYLK	CREATOR1	CREATOR2
Export	text ⁶ , DIF, SYLK	data	library
Notes	⁶ Can be delimited by commas, returns, tabs		

Product	OverVUE , Version: 2.0a	TYPE1
Company	<i>ProVUE Development Corporation</i>	DVSH
Category	database	OVUE
Data Types	text, number, date, variable	CREATOR1
Import	text ⁶ , DIF, SYLK, dBase II	
Export	text ⁶ , DIF, SYLK, dBase II	
Notes	⁶ Can be delimited by commas, returns or tabs	
Product	Quartet , Version: 1.0	TYPE1
Company	<i>Haba Systems</i>	QSAV
Category	spreadsheet	QRTO
Data Types	text, number	CREATOR1
Import		
Export		
Notes	Will not run on Macintosh Plus!	
Product	Rags to Riches Ledger , Version: 2.5 +	TYPE1
Company	<i>Chang Labs</i>	RRLD
Category	accounting	RTRL
Data Types	text, numbers	CREATOR1
Import	bitmap ⁵	
Export		
Notes	⁵ For decoration only, via Clipboard	
Product	ReadySetGo , Version: 2.0	TYPE1
Company	<i>Manhattan Graphics</i>	RSGI
Category	page layout	MART
Data Types	text, bitmap	CREATOR1
Import	text, bitmap ¹	
Export	bitmap ¹	
Notes	¹ Via Clipboard	

Product **StatView**, Version: 1.0
Company *Brainpower Inc.*
Category statistics
Data Types number
Import text, DIF, SYLK, WKS
Export text
Notes

TYPE1

FSTA

STAT

CREATOR1

normal

TYPE2

TEXT

STAT

CREATOR2

text

Product **ThunderScan**
Company *Thunderware Inc.*
Category hardware graphic input
Data Types bitmap
Import bitmap
Export bitmap
Notes

TYPE1

SCAN

SCAN

CREATOR1

Scan

TYPE2

PNTG

MPNT

CREATOR2

MacPaint

Product **Word**, Version: 1.05
Company *Microsoft*
Category word processor
Data Types text, bitmap
Import text, bitmap¹
Export text, bitmap¹
Notes ¹Via Clipboard

TYPE1

WDBN

WORD

CREATOR1

formatted

TYPE2

TEXT

WORD

CREATOR2

text only

APPENDIX



ASCII and EBCDIC





The ASCII Character Set


The American Standard Code for Information Interchange is the United States' adaptation of ITA-5. ITA-5 is short for International Telegraph Alphabet Number 5. The transnational organization that set this protocol in place is the International Telecommunications Union (ITU). The ITU is a civil (as contrasted with military) international network established to promote worldwide standards in telecommunications. Two permanent committees of the ITU contribute two other sets of acronyms you'll run across frequently as you delve into communications: the CCIR (International Consultative Committee on Radio) and the CCITT (International Consultative Committee on Telephone and Telegraph). You should also be aware of the International Organization for Standardization or ISO, because this body is always working on new standards that eventually affect international networking and computer information exchange on international phone and satellite links.

ITA-5 was jointly developed by the CCITT and the ISO and became the basis on which our American National Standards Institute (ANSI) came up with USASCII. If all that seems convoluted, it certainly is, but no one said standards, such as they are, come easily.

ASCII only defines seven of the eight bits usually used to transmit characters between computers. Therefore, different computer companies have been free to use the additional 128 symbols made possible by the eighth bit for their own internal purposes. In data transmission,

the eighth bit, also called the high-order bit or just the high bit, can be treated in a variety of ways by sender and receiver. It can be used as a parity bit, or it can be ignored, or it can be used to represent machine-specific characters such as the Macintosh graphic set or the IBM PC graphic set. ANSI has been developing a standard for the eighth-bit character set (called extended ASCII) since at least 1976, but so far none has been adopted. Overseas implementations of ITA-5 are not identical with the ASCII set.

The Macintosh keyboard can send all defined ASCII characters except NUL (ASCII 0). To send the control characters (ASCII codes 1–31) you hold down the  key while pressing the character to be sent. If you wanted to send a control-C (sometimes also indicated in print by ^ C) which is ASCII code 3, you hold down the  key and type C. It doesn't matter if the shift key is up or down when you do this. The ENTER key also sends a control-C. A carriage return is also control-M. The three most commonly used control keys in telecommunications are control-C (also known as the break character), control-S (X-Off, or pause), and control-Q (X-On, or resume). The ESC (escape) character (ASCII 27) is sent with the keyboard combination  [(right square bracket). Finally, the DEL (delete) character (ASCII 127) is generated using -backspace.

The break signal, control-C, should not be confused with the modem break signal, which is a longer tone recognized by some mainframe systems. On the Macintosh, this longer break signal must be specially generated by your terminal software and assigned either a menu function or an unused  key.

EBCDIC

The Extended Binary-Coded Decimal Interchange Code is an eight-bit code developed in the mainframe world by IBM for its System/360 machines. The 256 characters that comprise EBCDIC do not overlap ASCII at all.

For the most part, micro communications employ ASCII rather than EBCDIC. There is otherwise no clear superiority of one over the other. However, the sort order or collating sequence of the two systems is quite different. For example, a list sorted in ASCII order will not look the same as a list sorted in EBCDIC order. This could make a significant difference in some situations.

The following table is a combined ASCII/EBCDIC character-set reference. It shows the variations in stop bits and high-order bits.

Table B.1: Universal Code Chart/Hexadecimal Equivalency

[illegible]

- * The listing in the Baudot column may be in the form of two hex numbers, in which case the first number designates the **LETTERS/FIGURE SHIFT**. Where only one hex number is shown, the character is shift-independent.

APPENDIX



Sources Directory

Microcomputers

(See Appendix D for a brief catalog of Microcomputers.)

For a more comprehensive listing of micro, mainframe, and mini manufacturers, software publishing companies, and peripheral equipment manufacturers, see:

Data Sources

Ziff-Davis Publishing Company
One Park Avenue
New York, NY 10016
(609) 795-7012

Communications Accessories

Black Box Catalog

Black Box Corporation
Box 12800
Pittsburgh, PA 15241
Customer Tech Support: (412) 746-5565

This is a catalog of telecommunications products: modems, books, software, cables, connectors, and more—a telecom junky's heaven. The catalog also includes useful technical information and explanations of many basic telecommunications ideas. A gem of its kind! However, be aware that for a long time the Black Box people dealt only with "telecommunications professionals." They are slowly inching their way to the consumer market, especially with their modem and communications software offerings. If you need heavy support from your suppliers, be prepared to be patient with Black Box as they learn how to handle the rest of us non-technical people.

Telstar Computer Communications Catalog

Telstar Communications
145 Lincoln Street
Boston, MA 02111

A slick source for telecommunications accessories: modems, communications software, and so on. They're not comprehensive, nor are their prices the lowest. But the products they carry in the middle-to-high price range are all high quality and have excellent track records in the marketplace. Furthermore, the information they present in their product descriptions is also high quality. You can learn quite a bit just by browsing through their pleasingly printed catalog. Bear in mind, though, that you may find better software for your uses than what any catalog will carry, and as to price, shop around as you would for any consumer item.

Macintosh Plus Cable Supplies

The following companies supply Macintosh Plus cables and adapters. Note that not all of these suppliers provide a cable that includes the 5V power pin that Apple removed from its Macintosh Plus ports. These

look like DIN (European standard) connectors, but they are not. Apple calls them *mini-circular 8* connectors.

Apple Computer

System Peripheral-8 Cable PN: 590-0340A

Kette Group

8-to-9-Pin Adapters

MacNifty Central
6860 Shingle Creek Parkway
Minneapolis, MN 55430
(800) 328-0184

Apple Macintosh Plus Adapter Cable PN: 590-0341A

C Enterprises
310-110 Via Vera Cruz
San Marcos, CA 92069
(619) 744-8182

Molded 8-Pin Cable Assembly. AppleTalk Connectors.

H.B. Associates
P.O. Box 857
Union City, CA 94587
(800) 423-3014
(800) 423-4224 (in CA)

Phone Networks and Information Services

Starnet Long-Distance Service

This is an alternative long distance carrier, similar to MCI and Sprint, for those whose long distance bills run \$500 and up per month.

Starnet Corp.
3989 Ruffin Road
San Diego, CA 92123

TELENET**PC Pursuit**

A public-access data network, Telenet from GTE provides access to remote services and its own Telemail electronic mail service. PC Pursuit is a consumer/home-oriented communications service designed to provide micro-to-micro links on weekends and evenings. The service costs \$25 per month for an unlimited number of calls during the

established hours. As of this writing, PC Pursuit connects 12 major metro areas. GTE promises more over the coming months. You can register for the service by modem by calling (800) 835-3001.

GTE Telenet Communications Corp.
12490 Sunrise Valley Drive
Reston, VA 22096
(800) 368-4215 (Voice information)

GENie

Similar to PC Pursuit in the range of services offered, GENie is available through local phone service in 63 major cities in the U.S. GENie also provides a paper mail delivery service. You send your message electronically to the city of destination, and GENie prints it out and delivers it to anyone via the postal service.

General Electric Information Services
401 N. Washington Street
Rockville, MD 20850
(800) 638-9636

MCI

MCI Mail

MCI was the first independent competitor to AT&T when it started offering its long-distance call service in 1975. In 1983 it began to pitch microcomputer owners with its new electronic mail service, MCI Mail. It offers five methods of delivery:

1. The instant letter—from your MCI electronic mailbox to that of another MCI user (who also has a computer). The cost is forty-five cents for messages less than 500 characters long and one dollar for up to 7500 characters.
2. The MCI letter—transmitted to an MCI center from your computer. At the center, the letter is printed out and put into a regular paper envelope for delivery to the addressee, usually within two days of transmission, via the regular U.S. postal service. The cost is two dollars each.
3. The overnight letter—same as above, only faster paper, for eight dollars a throw.
4. A 4-hour letter—same as the last two, only hand-delivered for about thirty dollars.
5. A Telex address—worldwide, about 1,000,000 companies use Telex for their electronic mail. The Telex machine prints a hard copy for in-house delivery through the internal company mail system.

MCI charges no monthly minimum. Instead, there is an annual fee (\$18 as of this writing). The company has also begun to offer a special bulletin board service which charges a per-hour connect fee.

MCI Communications Corp.
1133 19th Street NW
Washington, DC 20036
(800) MCI-2255

Western Union

Easy Link

Easy Link is aimed at micro users within businesses. It offers access to the Telex network for fifty cents per minute. The fee structure is a bit more intricate than MCI's. There are a variety of billing plans depending on company needs and preferences, transmission speeds, and so on. Mailgrams can be sent (three dollars for the first page, seventy-five cents per page thereafter). Cablegrams (international messages to non-Telex users) are available. Overnight letters cost \$7.75 for a message of up to five pages, twenty-five cents for each additional page. Finally, a two-hour hard-copy letter can be sent to anyone in 30 American cities (\$20 up to five pages, fifty cents each additional page) and can be sent worldwide overnight.

Western Union
1 Lake Street
Upper Saddle River, NJ 07458
(201) 826-5000

The Source

The Source focuses on consumer services, such as special interest forums (electronic conferencing) and airline schedules. There is a \$49.95 "initiation fee" (sometimes waived during special promotions); thirty-six cents per minute at 300 bps, prime time; forty-three cents a minute at 1200 bps, prime time; forty-six cents per minute at 2400 bps, prime time; nonprime-time rates are fourteen, eighteen, and twenty cents per minute.

The Source also lets subscribers send printed-out messages to non-Source subscribers via Western Union Mailgram.

Source Telecomputing
1616 Anderson Road
McLean, VA 22102
(703) 821-8888

CompuServe

CompuServe is the home of MAUG™ The Micronetworked Apple Users Group. This special-interest network that has grown up around CompuServe is a great source for Macintosh public-domain programs, technical information, and gossip. We are not sure if MAUG alone is worth the price of a CompuServe subscription, however. Much of the information is also available from other sources, including publically accessible bulletin board systems and local user groups. There is a \$39.95 "initiation fee" (sometimes waived during special promotions). Other fees are \$12.50 per hour (prime time); \$6.00 per hour evenings and weekends; \$15 and \$12.50 (respectively) at 1200 bps.

CompuServe, Inc.

5000 Arlington Centre Boulevard

Post Office Box 20212

Columbus, OH 43220

(800) 848-8199

(614) 457-0802

BRS (Bibliographic Reference Service)**BRS After Dark****BRKTHRU**

BRS offers access to business and industry data bases. Disciplines covered include science and technology, the biosciences, medicine, finance, education, the social sciences, and humanities. There are two main services from BRS: After Dark and BRKTHRU. The former, as its name implies, is available only in the evening hours. The latter is available 24 hours.

BRS Information Technologies

1200 Route 7

Latham, NY 12110

(800) 2-ASKBRS

(800) 227-5277

(800) 553-5566

DIALOG**Knowledge Index**

This service is for heavy-duty business, scientific, and professional research. Knowledge Index is a consumer-oriented selection of data bases. The full Dialog service offers many more. There is no initiation fee; however, access to individual data bases is charged according to

the rates set by the data base providers. This ranges from a low of \$20 per hour of connect time, to a high of \$300 per hour and up, depending on what data base is being searched. In addition, Dialog charges a connect charge of \$6 per hour.

Dialog Information Services
3460 Hillview Avenue
Palo Alto, CA 94304
In California: (800) 982-5838
In Canada: (416) 593-5211
In the U.K.: 0865-730969

Manufacturers Mentioned in the Text

Abaton Drive 5.25

Abaton Technology Corporation
7901 Stoneridge Drive, Suite 500
Pleasanton, CA 94566
(415) 463-8822

MacMail

Aegis
2210 Wilshire Boulevard #277
Santa Monica, CA 90403
(213) 392-6445

Tempo

Affinity Microsystems, Ltd.
1050 Walnut Street
Suite 425
Boulder, CO 80302
(303) 442-4840

Pagemaker

Aldus Corporation
411 First Avenue, South, Suite 200
Seattle, WA 98104
(206) 622-5500

MacDraw
AppleTalk
LaserWriter
MacPaint
MacWrite
MacTerminal

Apple Computer Inc.
20525 Mariani Avenue
Cupertino, CA 95014
(408) 996-1010

Home Accountant

Arrays, Inc./Continental Software
11223 South Hindry Avenue
Los Angeles, CA 90045
(213) 410-3977

dBASE

Ashton-Tate
10150 W. Jefferson Boulevard
Culver City, CA 90230

MacMainframe

Avatar Technologies, Inc.
99 South Street
Hopkinton, MA 01748
(617) 435-6872

Omnis 3

Blyth Software, Inc.
2929 Campus Drive, Suite 425
San Mateo, CA 94403
(800) 843-8615

Sidekick

Borland International
4585 Scotts Valley Drive
Scotts Valley, CA 95066
(408) 438-8400

StatView

Brainpower, Inc.
24009 Ventura Boulevard, Suite 250
Calabasas, CA 91302
(818) 884-6911

Softstrip

Cauzin Systems, Inc.
835 South Main Street
Waterbury, CT 06706
(800) 533-7323

TOPS

Centram
2560 Ninth Street
Suite 220
Berkeley, CA 94710
(415) 549-5900

Rags to Riches Ledger

Chang Labs, Inc.
5300 Stevens Creek Boulevard
San Jose, CA 95129-1088
(408) 246-8020

Kermit

Columbia University Center
7th Floor, Watson Laboratory
612 West 115th Street
New York, NY 10025
(212) 280-3555

BLAST

Communications Research Group
5615 Corporate Boulevard, Third Floor
Baton Rouge, LA 70808
(504) 923-0888

MacLink

(Two versions: one with cable, one without cable.)

DataViz, Inc.
16 Winfield Street
East Norwalk, CT 06855
(203) 866-4944

MacCharlie

Dayna Communications, Inc.
50 South Main Street, Suite 530
Salt Lake City, UT 84144
(801) 531-0600

PC to Mac and Back

Dilithium Press
921 South West Washington, Suite 870
Portland, OR 97205
(800) 547-1842

**Straight Talk
Market Manager Plus**

Dow Jones & Co., Inc.
P.O. Box 300
Princeton, NJ 08540
(800) 257-5114

Quick and Dirty Utilities

Dreams of the Phoenix, Inc.
P.O. Box 10273
Jacksonville, FL 32247
(904) 396-6952

Easy 3-D

Enabling Technologies, Inc.
600 South Dearborn, Suite 1304
Chicago, IL 60605
(312) 427-0408

FileMaker

Forethought, Inc.
250 Sobrante Way
Sunnyvale, CA 94086
(408) 737-7070

Quartet

Haba Systems, Inc.
6711 Valjean Avenue
Van Nuys, CA 91406
(800) FOUR-HABA (CA)
(800) HOT-HABA

**Art Grabber with Body Shop
Ensemble**

**Smartcom II
MicroModem II
Smartmodem**

Hayden Software Company
600 Suffolk Street
Lowell, MA 01854

Hayes Microcomputer Products, Inc.
P.O. Box 105203
Atlanta, GA 30348
(404) 449-8791

MacDraft

Innovative Data Design, Inc.
1975 Willow Pass Road, Suite 8
Concord, CA 94520
(415) 680-6818

SmartCable

IQ Technologies
11811 N.E. First Street, Suite 308
Bellevue, WA 98005
(206) 451-0232

Jazz

1-2-3

Lotus Development Corporation
55 Cambridge Parkway
Cambridge, MA 02142
(617) 577-8500

**Telescope Terminal Software
N'Cryptor**

Mainstay
5311B Derry Avenue
Agoura Hills, CA 91301
(818) 991-6540

ReadySetGo

Manhattan Graphics
401 Columbus Avenue
Valhalla, NY 10595
(914) 769-2800

**Excel
Chart
Multiplan**

Microsoft Corporation
16011 NE 36th Way
Box 97017
Redmond, WA 98073-9717
(206) 882-8080

Crosstalk Terminal Software

Microstuf, Inc.
1845 The Exchange, Suite 140
Atlanta, GA 30339
(404) 952-0267
(800) 241-6393

Magic Video Digitizer

New Image Technology
10300 Greenbelt Road, Suite 104
Seabrook, MA 20706
(301) 464-3100

SmartCat Modem

Novation, Inc.
18664 Oxnard Street
Tarzana, CA 92356
(213) 996-5060

Helix

Odesta Corporation
4084 Commercial Avenue
Northbrook, IL 60062
(800) 323-5423

inTalk

Palantir Software
12777 Jones Road, Suite 100
Houston, TX 77070
(800) 955-3797

Mach v1.0

Palo Alto Shipping Company
P.O. Box 7430
Menlo Park, CA 94026
(800) 44-FORTH

VersaTerm

Peripherals, Supplies and Computers
215 Mt. Penn Avenue
Perkiomen, PA 19606
(215) 779-0522

PopCom Modem

Prentice Corporation
266 Caspian Drive
P.O. Box 3544
Sunnyvale, CA 94088-3544
(408) 734-9810

Crunch

Paladin Software Corporation
3255 Scott Boulevard
Building 7, Suite C
Santa Clara, CA 95054
(408) 970-7300

Pretty Good Terminal

Phil's Pretty Good Software
440 South 45th Street
Boulder, CO 80303
(303) 499-5179

OverVUE

ProVUE Development Corporation
222 22nd Street
Huntington Beach, CA 92648
(714) 969-2431

Racal-Vadic Modems**Maxwell Modem****MacGeorge**

RD Communications, Inc.
1525 McCarthy Boulevard
Milpitas, CA 95035
(408) 946-2227

Interlace

Singular Software
4585 Scotts Valley Drive
Scotts Valley, CA 95066
(408) 438-8400

MicroPhone

Software Ventures Corporation
2907 Claremont Avenue, Suite 220
Berkeley, CA 94705
(415) 644-3232

Software Bridge

Systems Compatibility Corporation
One East Wacker Drive, Suite 1320
Chicago, IL 60601
(312) 329-0700

ClickArt Special Effects**ClickOn Worksheet**

T/Maker Graphics
2115 Landings Drive
Mountain View, CA 94043
(415) 962-0195

FileVision

Telos Software Products
3420 Ocean Park Boulevard
Santa Monica, CA 90405-6322
(213) 450-2424

Tektronix Terminals

Tektronix, Inc.
Box 500
Beaverton, OR 97077
(503) 627-7111

ThunderScan

Thunderware, Inc.
21 Orinda Way
Orinda, CA 94563
(415) 254-6581

Red Ryder

The FreeSoft Company
10828 Lacklink
St. Louis, MO 63114
(314) 423-2190

Trailblazer Modem

Telebit Corporation
10440 Bubb Road
Cupertino, CA 95014
(408) 996-8000

ASCII Express: The Professional

United Software Industries, Inc.
8399 Topanga Canyon Boulevard, Suite 200
Canoga Park, CA 91304
(818) 887-5800

Autolink 212A Modem

U.S. Robotics
8100 McCormick Avenue
Skokie, IL 60077
(312) 982-5010

Directories of On-line Information Sources

Directory of On-Line Data Bases (print)

Cuadra Associates, Inc.
2001 Wilshire Boulevard
Suite 304
Santa Maria, CA 90403
(213) 829-9972

The Computer Data and Data Base Source Book (print)

by Matthew Lesko
Avon Books
1790 Broadway
New York, NY 10019

Datapro Directory of On-Line Services (print)

DataPro Research Corporation
1805 Underwood Boulevard
Delran, NJ 08075
(609) 764-0100

Databasics: Your Guide to On-Line Information (print)

Garland Publishing
136 Madison Avenue
New York, NY 10016
(212) 686-7492

Data Base Update (print)

10076 Boca Entrada Boulevard
Boca Raton, FL 33433
(800) 345-8112

Data Base Alert (print)

Knowledge Industry Publications
701 Westchester Avenue
White Plains, NY 10604
(800) 431-1880

Instant Yellow Page Service (electronic)

P.O. Box 27347
Omaha, NE 68127
(402) 331-7169

Work-at-Home and Freelance Consultant Information

Telecommuting Review

TeleSpan Publishing Corporation
50 West Palm Street
Altadena, CA 91001

Computer Entrepreneur

P.O. Box 456
Grand Central Station
New York, NY 10136

National Association for the Cottage Industry

P.O. Box 14460
Chicago, IL 60614

Personal Media Newsletter

CommuniTree Group
1047 Sutter Street
San Francisco, CA 94109

Government Information

NTIA—The National Telecommunications and Information Administration

NTIA is a body set up to “coordinate Federal Government activities involved in new information technologies.” How well it does so is debated among experts. However, it is a source of a lot of technical data. Whether or not it brings “a cohesive foundation to our electronic future,” NTIA pronouncements and publications are worth monitoring, since it is supposed to act as the President’s principal advisor on telecom matters.

U.S. Department of Commerce
NTIA Public Information Office
1800 G Street NW, Room 774
Washington, DC 20504
(202) 377-1832

Congressional Quarterly

Anyone interested in following the course of legislation through Congress can use this on-line service. The data base includes information on the course of bills through Congress, committees, and interpretive news articles. It's "pricey" at \$24 per 15 minutes of connect time, but for those whose livelihoods or specialties demand it, the information is up-to-the-minute and comprehensive.

Congressional Quarterly, Inc.
1414 22nd Street NW
Washington, DC 20037
(202) 887-8511

Additional Special Use Software

PC MacBridge

Software/hardware package lets you put an IBM PC on the AppleTalk personal network. Plug-in board fits PCs and most clones. Company also supplies software for converting PC files for compatibility with laser printers.

From:
Tangent Technologies, Ltd.
5720 Peachtree Parkway
Norcross, GA 30092
(404) 662-0366

Tekalike A Graphic Terminal

Tekalike software turns the Macintosh into a Tektronix-compatible terminal. The software package supports all the standard Tektronix 4014 commands (also 4006, 4010, 4012, and 4016). This package is compatible with many on-line service software packages. This package is useful for previewing mainframe graphics with the Macintosh, and integrating mainframe graphics capabilities with the Macintosh software. It will convert mainframe graphics into MacPaint or MacDraw documents for use in many Macintosh applications (MacWrite, Page-maker, and so on).

From:
Mesa Graphics
P.O. Box 600
Los Alamos, NM 87544
(505) 672-1998 Telex: 5101003099

Conference Trees

Conference Tree host software runs on the MS DOS family of micros (see Appendix D). PC TeleTree supports small- to medium-sized work groups with electronic conferencing, private message exchange, and DOS file exchange. It will also support Macintosh file and application exchange in conjunction with BinHex translation. Low maintenance overhead, unattended operation, quick set up, and short training times mark this package for use in industry and academia.

From:
Synergetic Communications
Box 9964
Berkeley, CA 94709
(415) 548-8170

Special Interest Networks (SPINs)

An on-line data base and conferencing system organized and run by Gary Hart is available to all micro users.

Contact:
Center for a New Democracy
220 I Street NE
Suite 220
Washington, DC 20002
(202) 675-6050 (voice)

Investor's Workshop

(702) 438-1398
300/1200 baud

Stock, bond, and commodity investment information and expertise exchange. No buying or selling of same. Additional financial soft-

ware and on-line information is available in *The Wall Street Software Directory* from:

The Grinde Group Limited
Route 198
Woodstock Valley, CT 06282
(203) 974-2227

International Association of Cryptologic Research

Cryptology, the science of making and breaking secret codes, is the topic of this on-line bulletin board service. (703) 237-4322. 300/1200 bps, 24 hours on weekends, 6:30 pm to 7:00 am on weekdays.

NASA's BBS

An on-line forum for space fans. (301) 344-9156. 300 bps only.

APPENDIX



A Brief Catalog of Micro and Terminal Families

This catalog of microcomputers and terminals is grouped by family. It is not intended to be a complete listing. For more comprehensive sources, you should turn to a directory such as *DATA Sources* (see Appendix C).

A caveat: just because a microcomputer is in the MS DOS category (or any other for that matter) does not mean that it will behave the same as another computer in the same family under all the same conditions. This fact can be distressing, especially when people think they have bought an “XYZ-compatible machine” only to find out that many programs written for the XYZ will not work with their new machine.

Communications difficulties between families may be compounded because of these basic intra-family dissimilarities. Don't be discouraged if you have followed all the guidelines found in *MacAccess* only to find that they don't work with the particular brand you're trying to connect to.

By and large, all the machines do use the RS-232 serial interface (at least some version of it) and persistent trial and error will eventually get results. If all else fails, call the customer support department of the manufacturer.

Macintosh Family

Macintosh (128-512K) and Macintosh Plus (1024K) Macintosh XL with MacWorks

Apple Computer
20525 Mariani Avenue
Cupertino, CA 95014
(408) 996-1010

PC DOS / MS-DOS Family

ADDS PC/I ADDS PC/II

ADDS Inc.
Applied Digital Data Systems, Inc.
100 Marcus Boulevard
Hauppauge, NY 11788
(516) 231-5400

Alpha Micro ELS The Workstation

Alpha Microsystems
17322 Von Karman
P.O. Box 18347
Irvine, CA 92713
(714) 957-8500

AJ Passport MBA AJ Passport PHD

Anderson Jacobson, Inc.
521 Charcot Avenue
San Jose, CA 95131
(408) 263-8520

Apricot F1
Apricot PC
Apricot Portable
Apricot Xi, Apricot Xi 20

Apricot, Inc.
3375 Scott Boulevard
Suite 336
Santa Clara, CA 95051
(408) 727-8090

PC 6300

AT&T Information Systems
1 Speedwell Avenue
Morristown, NJ 07960
(201) 898-2000

Personal Computer A-200

Canon USA, Inc.
1 Canon Plaza
Lake Success, NY 11042
(516) 488-6700

Colby PC3

Colby Computer
849 Independence Avenue
Mountain View, CA 94043
(415) 968-1410

MPC and VP Series

Columbia Data Products, Inc.
9150 Rumsey Road
Columbia, MD 21045
(301) 992-3400

**Compaq
Compaq Plus
Deskpro
Compaq Portable, Portable II**

Compaq Computer Corporation
20333 FM 149
Houston, TX 77070
(713) 370-7040

PC 400

Corona Data Systems
275 East Hillcrest Drive
Thousand Oaks, CA 91360
(805) 495-5800

**Desktop Generation Models 10, 10/SP
Data General One**

Data General Corporation
4400 Corporate Drive
Westboro, MA 01581
(617) 366-8911

M-18 Desktop & Portable

Docutel/Olivetti
5615 Highpoint
Irving, TX 75062
(214) 258-5400

**Eagle PC Plus Series
Eagle PC SPIRIT Series
TURBO Series**

Eagle Computer, Inc.
7100 Chapman Avenue
Garden Grove, CA 92641
(714) 891-2665

**Touchscreen
Touchscreen MAX
HP-110**

Hewlett-Packard
11000 Wolfe Road
Cupertino, CA 95014
(800) FOR-HPPC

**IBM PC, PC/XT, PC AT
Portable PC
PCjr**

IBM Corporation
P.O. Box 1328
Boca Raton, FL 33432
(305) 998-6607

ITT XTRA

ITT Information Systems
2041 Lundy Avenue
San Jose, CA 95131
(408) 945-8950

Kaypro 16

Kaypro Corporation
533 Stevens Avenue
Solana Beach, CA 92075
(619) 481-4300

Leading Edge Personal Computer

Leading Edge Products, Inc.
225 Turnpike Street
Canton, MA 02021
(800) 343-6833

Sr. Partner

Panasonic Industrial Company
One Panasonic Way
Secaucus, NJ 07094
(201) 348-7000

Sperry Personal Computer**Sperry Portable Computer**

Sperry Corporation/Computer Systems

P.O. Box 500

Blue Bell, PA 19424

(215) 542-4011

Model 1000**Model 1200HD****Model 2000 & 2000HD**

Tandy-Radio Shack

1300 One Tandy Center

Fort Worth, TX 76102

(817) 390-3011

Apple II Family

Apple II, II+, IIe, IIfx

Apple Computer, Inc.

20525 Mariani Avenue

Cupertino, CA 95014

(408) 996-1010

Apricot PC

Apricot, Inc.

3375 Scott Boulevard

Suite 336

Santa Clara, CA 95051

(408) 727-8090

Terminal Families Chart

(Courtesy of Palantir Software)

These terminals are grouped by type. We do not provide manufacturers' names and addresses due partly to space limitations, and partly to the fact that, generally, the manufacturers of these terminals won't be that helpful in cases where you are trying to emulate one of their terminals rather than actually using one. We suggest that you enlist the aid of the person or persons who operate the particular mainframe or mini with which you are trying to connect.

ADDs Viewpoint 60

[used on NCR systems and systems running PICK]

ADDs Viewpoint and Regent Series
CTi Data 3078 (in ADDs VP60 mode)
Televideo 914 (in ADDs VP A-2 mode)

Beehive DM5A

[used on Harris and CROMEMCO systems]

Cromemco 3101, 3102 and C-10
Beehive DM5 series, Topper, ATL-78, ATL-004
Beehive terminals in VT100 emulation mode use VT-100 instead

IBM 3101

Televideo 925

Esprit Systems, Esprit, Esprit II, Esprit III
Interaction systems TT150 (in Televideo 950 mode)
Lear Siegler ADM3A, ADM5
Liberty Electronics Freedom 100, 110, 200 (in Televideo mode)
Morrow MDT-60
Paradyne 7811-01
Qume QVT-102, 108 (in Televideo mode)
Soroc Challenger
Televideo 910, 910, 912, 914, 920, 924, 925, 950, Personal Terminal
Visual V500 (in LSI ADM3A mode)
Wyse 100, 300, 50
Zenith Z-29 (in Lear Siegler ADM-3A mode)

VIDTEX

[Exclusive use with CompuServe to display graphics]

VT100 (ANSI)

[Used on DEC computers and emulated by many others]

Altos II
Ann Arbor Ambassador; Genie; Guru
ANSI
CIE Terminals CIT-101, 101e, 161, 467, 500, 80
Colorgraphics Communications MVI-100, 489, 119, 11, 813, 819, MVI-7,
719
DEC VT100, 101, 102, 131, 125, 220, 240, 241 (except graphics)
FALCO Fame 100, Fame 78, Fame 3, TS-1, TS-100/132
General Terminal SW 10 (in ANSI mode)
GraphOn GO-100, GO-140

Human Designed Systems Concept AVT series
Ithaca Intersystems Graphos I, II, III
InterColor 2405, 2427, VHR19
Kimtron KT-10
NCR 7910 (in ANSI mode)
Prime PST100
TAB 132/15
Tandberg Data TDV 2220
TEC ET 80B
Tektronix 4105, 4107, 4109 (in ANSI mode only, no graphics)
Teleray Model 100, 16, 7
Teletype 5410, 5420
Televideo 970
Vector Graphics Vector 4, 4S, VSX
Visual Technology V102, 100, 300
Wyse 75
Zenith Z-19-HW, Z-29, Z-49

VT52

[Older DEC terminal widely emulated by others. Most terminals listed under VT100 also emulate VT52]

Tymshare Scanset (in VT52 mode)
Visual V330 (in VT52 mode)
Zenith Z-10, Z-19-HW (in VT52 mode)

APPENDIX



Two Softstrip® Data Strips

Softstrips are a new, inexpensive way of distributing computer software on paper. The strips may look like a piano roll unrolled but are, in fact, two programs to run on your Macintosh. We think you'll find them useful in your telecom work.

If you don't know about Softstrip readers—the devices that are used to actually enter the programs into your Macintosh—you can find out more by writing or calling:

Cauzin Systems, Inc.
835 South Main Street
Waterbury, CT 06706
(203) 573-0150

Both of these programs are reproduced here with the kind permission of their authors, Yves Lempereur and Ken Winograd.

BinHex 5.0

The first Softstrip (Strip 1) contains a copy of BinHex 5.0, which is the latest version as *MacAccess* goes to press. You can use this to convert

outgoing files for transmission to another non-Macintosh system, and to reconvert files that you capture. (See “Macintosh to Macintosh Through an Intermediate System” in Chapter Three.) Note that this is a shareware product. If you find it useful, the author requests that you send \$10 and include your name and address so that you can be informed about future developments. Make check payable to Yves Lempereur and send to:

28611B Canwood Street
Agoura Hills, CA 91301
(818) 991-6540

Instructions: Read the data strip into your Macintosh. Put it with your telecom program or telecom utilities. When you’re ready to convert a file using BinHex, just double-click on the BinHex 5.0 icon from the desktop. Follow the instructions from there.

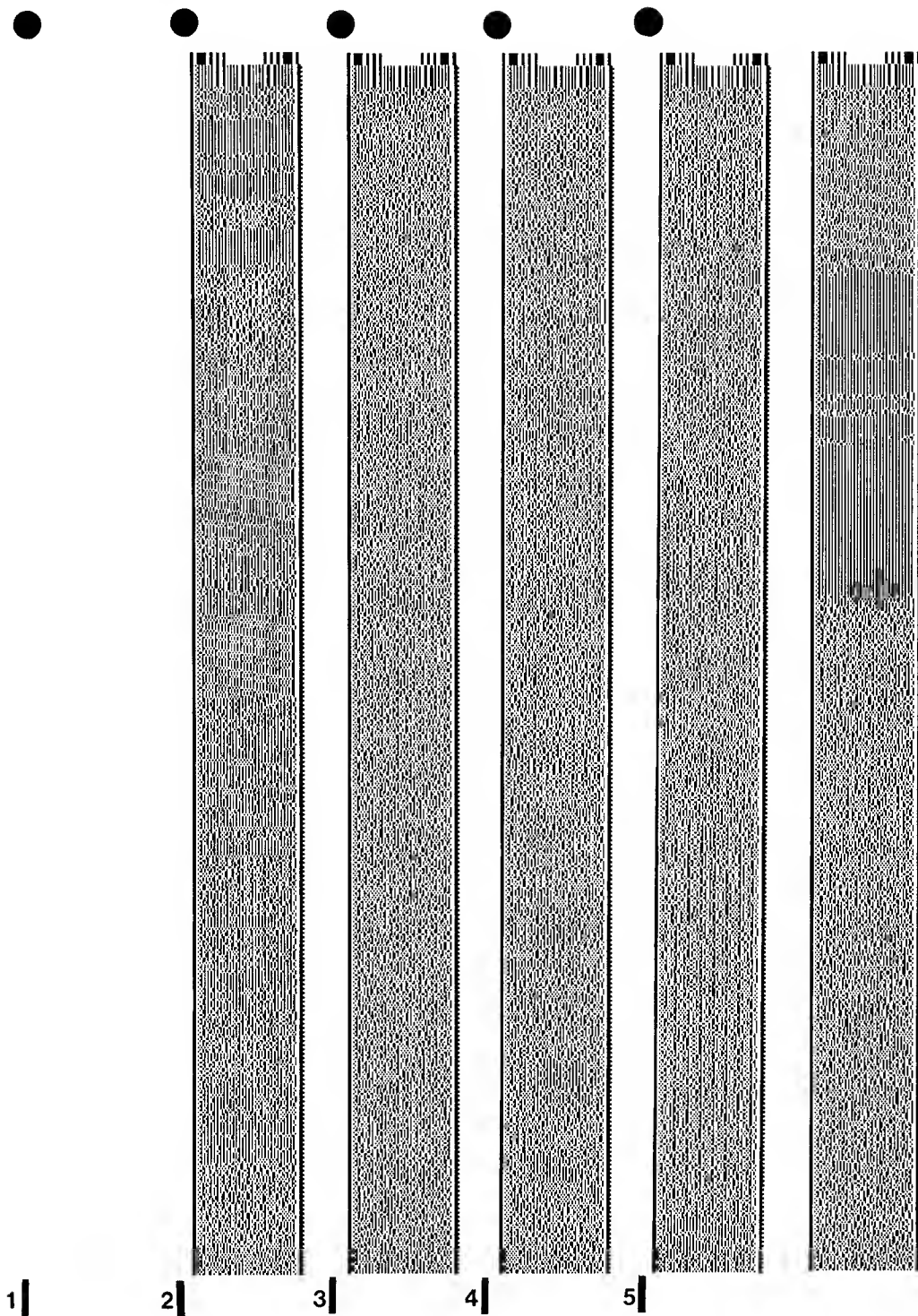
DAFile

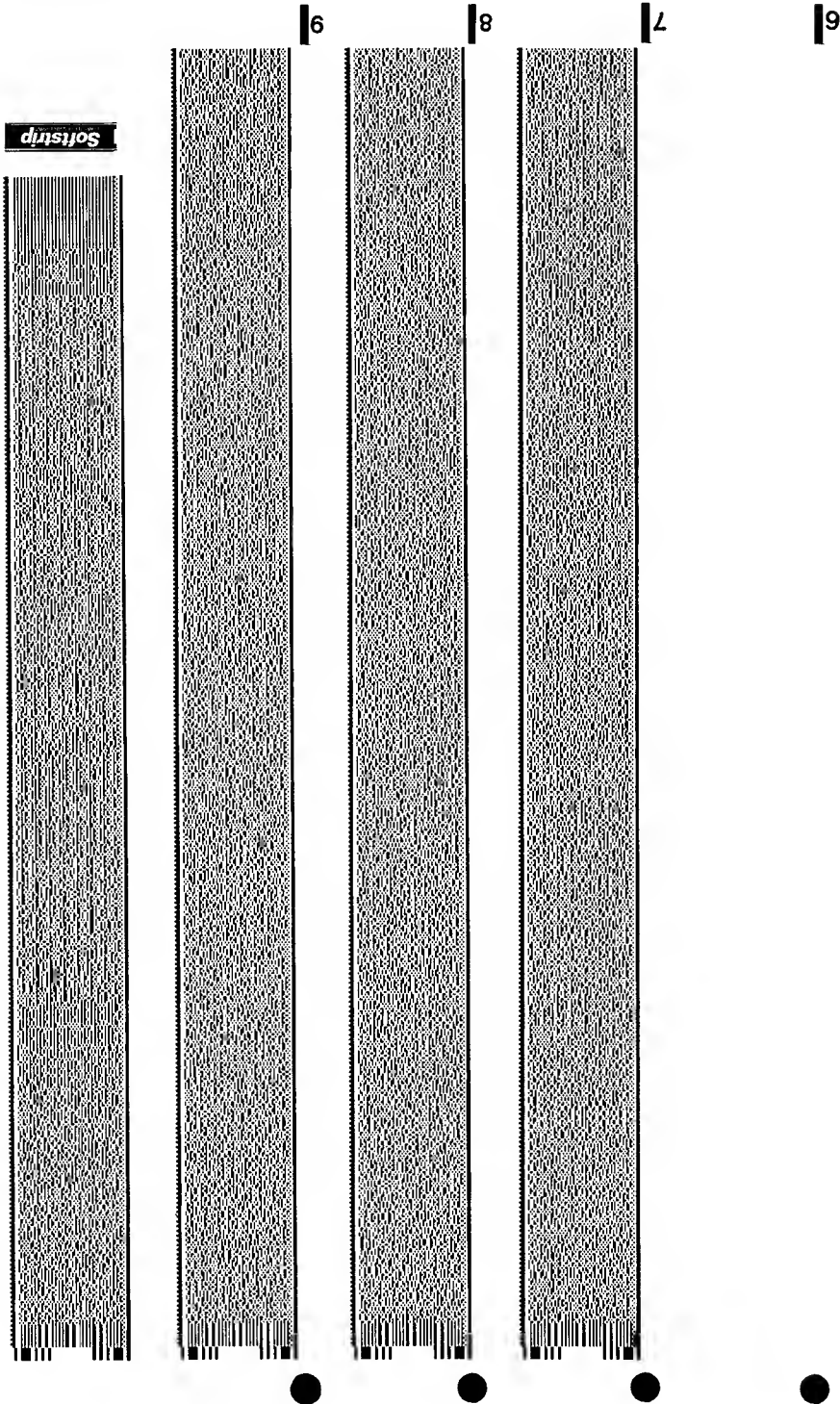
The second Softstrip (Strip 2) is a copy of another shareware accessory that comes in handy when dealing with files and file transfers. DAFile, by Ken Winograd, lets you look at various characteristics of your Macintosh files.

If you find this accessory to be as useful as we say, the author asks \$5-\$10 for its use. Make check payable to Ken Winograd and send to:

2039 Country Club Dr.
Manchester, NH 03102

Instructions: Read the data strips into your Macintosh. Then use Font/DAMover to install this desk accessory in your Apple menu. When you want to scope out a file, pull down the Apple menu and select DAFile. Happy sleuthing!





APPENDIX



Modem Commands and Result Codes

The following tables lists the commands and result codes for the Apple Personal Modem and Hayes Smartmodem 1200. Note that many other “Hayes-compatible” modems use similar—*but not identical*—command sets and result codes. Consult your modem manual for details. Boldface indicates default settings (these are the settings you get when you turn on the modem or reset the modem with ATZ).

Table F1 Apple Personal Modem and Hayes Smartmodem 1200
Command Sets

Command	Meaning
AT	Attention (precedes all commands except A/ and + + +)
A/	Repeat last command (do not follow by Return)
+ + +	Escape—go from on-line state to command state (1-second pause needed before and after code; do not follow by Return)
.....	
!	(see D command)
,	(see D command)

Table F1 (cont)

Command	Meaning
/	(see D command)
;	(see D command)
@	(see D command)
A	Answer call without waiting for ring; switch from voice to computer communications
<u>BQ</u>	<i>Apple</i> : Normal transmitter level (-10dB) <i>Hayes</i> : CCITT V.22 mode
B1	<i>Apple</i> : Decrease transmit level by 3dB (to compensate for PBX or CBX) <i>Hayes</i> : Bell 103 and 212A mode
C0	Transmit carrier off
C1	Transmit carrier on
D	Dial number that follows
D ...!	<i>Hayes only</i> : Inserts "flash" command to transfer a call
D ...;	Pause in dialing (length of pause set by S8)
D .../...	<i>Hayes only</i> : Wait for 1/8 second
D ...;	Return to command mode after dialing (always last character)
D ...@...	<i>Hayes only</i> : Wait for silence
DP	Dial pulse
D ...R	Call to "originate-only" modem (goes at end of number)
DT	Dial touch-tone
DW	<i>Hayes only</i> : Wait for second dial tone
E0	Local echo off in command mode
E1	Local echo on in command mode
F0	Half duplex (turns local echo on in data mode)
F1	Full duplex (turns local echo off in data mode)
H0	Hang up (remember to use + + + to enter command mode)
H1	Take phone off hook
H2	<i>Hayes only</i> : Take phone off hook (line relay only)

Table F1 (cont)

Command	Meaning
I0	<i>Apple</i> : Display product code (first two digits show modem speed, the last digit gives product revision number) <i>Hayes</i> : Display product code (first two digits show product, the right digit shows product revision number)
I1	Displays ROM checksum (manufacturer's use only)
I2	<i>Hayes only</i> : Test internal memory
M0	Speaker off
M1	Speaker on for command mode only
M2	Speaker always on
O	Return to on-line (data) mode
O1	<i>Hayes only</i> : Remote digital loopback off
O2	<i>Hayes only</i> : Remote digital loopback request
P	(see D command)
Q0	Display result codes
Q1	Quiet—result codes not displayed
R	(see D command)
S0 = n	<i>Apple</i> : Answer phone on n^{th} ring (0–255; default 0) <i>Hayes</i> : Answer phone on n^{th} ring (default set by switch 5)
S1	Count number of rings (up to 255)
S2 = ASCII	Escape character (ASCII code from 0–127; default 43)
S3 = ASCII	End-of-line character (ASCII code from 0–127; default 13)
S4 = ASCII	Line feed character (ASCII code from 0–127; default 10)
S5 = ASCII	Backspace character (ASCII code from 0–32, 127; default 8)
S6 = secs	Wait for dial tone (2–255; default is 2)
S7 = secs	<i>Apple</i> : Wait for carrier (1–255; default is 30) <i>Hayes</i> : Wait for carrier (1–60; default is 30)
S8 = secs	Pause for comma while dialing (0–255; default 2)
S9 = .1secs	<i>Apple</i> : Carrier detect response (not adjustable—set at 6) <i>Hayes</i> : Carrier detect response (1–255; default 6)

Table F1 (cont)

Command	Meaning
S10 = .1secs	Disconnect timing (1-255; default 7)
S11 = msec	Touch-tone speed (50-255; default 70)
S12 = .02 secs	Escape code guard time (20-255; default 50)
S13	<i>Apple only:</i> UART status register (bit-mapped)
S14	<i>Apple only:</i> Option register (bit-mapped)
S15	<i>Apple only:</i> Flag register (bit-mapped)
S16 = n	0 = Turn off self test 1 = Turn on self test
T	(see D command)
V0	Transmit result messages as code numbers
V1	Transmit result messages as words
W	(see D command)
X0	<i>Apple:</i> Use result codes 0-4 <i>Hayes:</i> Compatible with Smartmodem 300
X1	<i>Apple:</i> Use result codes 0-5 <i>Hayes:</i> Enable result code CONNECT 1200
X2	<i>Apple:</i> Use result codes 0-12; turn on continuous redial <i>Hayes:</i> Enable dial tone detection
X3	<i>Hayes only:</i> Enable busy signal detection
X4	<i>Hayes only:</i> Enable dial tone and busy signal detection
Z	Reset to modem default settings

Table F2 Apple Personal Modem and Hayes Smartmodem 11200 Result Codes

Value	Message	Meaning
0	OK	Command executed
1	CONNECT	<i>Apple:</i> Carrier signal detected; going on line <i>Hayes:</i> Connected at 300 or 1200 bps; connected at 300 bps is result of X1, X2, X3 or X4 command
2	RING	<i>Apple:</i> Ring signal detected; enter Answer Mode <i>Hayes:</i> Ring signal detected; answer call if auto-answer mode or ATA command given
3	NO CARRIER	Carrier signal not detected, or lost
4	ERROR	<i>Apple:</i> Invalid command, or 60-character memory buffer exceeded <i>Hayes:</i> Invalid command, 40-character memory buffer exceeded; cannot operate at 300 bps in CCITT V.22 mode; invalid character format at 1200 bps; invalid checksum
5	CONNECT 1200	Connected at 1200 bps
6	NO DIAL TONE	Dial tone not detected within allotted time
7	BUSY	Busy signal detected
8	NO ANSWER	<i>Hayes only:</i> Silence not detected (from @ command)
11	RINGING	<i>Apple only:</i> Line you are calling is ringing (i.e., not busy)
12	VOICE	<i>Apple only:</i> Line you called has been answered by a person

Glossary



[M] designates a Macintosh term.

acoustic coupler—A type of modem designed to transmit and receive data through a telephone handset. The handset is placed in a cradle consisting of two rubber cups, one for the mouthpiece and one for the earpiece. An acoustic-coupled modem is sometimes called a data set. Such a modem must be used when it is not possible to plug a direct-connect modem into a telephone jack, as in some offices and most hotel rooms.

alphanumeric—Used to describe a character that can be a number, letter of the alphabet, or a special symbol

ANSI—(American National Standards Institute). An official body that publishes standards in computing and related fields, such as computer languages

AppleTalk connector—A small box with a short cable that connects with a computer, printer, or other device. The connector box contains two 3-pin connectors that hook up with the AppleTalk network cables.

ASCII file—A file containing alphanumeric characters (text) and the control characters defined by the American Standard Code for Information Interchange (ASCII). ASCII files are created by using the Text Only option (if available) when saving Macintosh word processing documents on disk.

asynchronous communication—The kind of data transmission used by most personal computers, including the Macintosh. Both ends of the transmission must be set to the same speed, but they needn't keep in perfect step, which would require some kind of clock signal. Asynchronous communica-

tion relies on start and stop bits to inform the computers at either end when transmission of characters is initiated and completed.

autoanswer—A modem feature that describes the fact that the modem can pick up the phone line when a ring signal is detected and send out the answer tone without having to be told (by you or the software) to do so

autodial—A modem feature that describes the modem's ability to dial a phone number. You type in the number at your keyboard, and the modem dials. In earlier days, you had to dial the number on a regular phone, then switch in the modem. Having the autodial feature usually means that you can put the dialing of the phone under software control.

baud—A unit of measure for data transmission speed. The speed in baud is the number of discrete conditions or signal events each second. When each signal event represents only one bit, then the baud is identical to the number of bits per second (bps).

Bell-compatible—A modem is said to be Bell-compatible when the audio tones issued by the modem meet Bell Telephone standards. Variants of the term include Bell 103-Compatible (standard for 300-baud modems) and Bell 212A-Compatible (standard for full-duplex, 1200-baud modems).

BIT (binary digit)—One of two digits that represent data in a form suitable for use by computers. All data can be reduced to binary digits.

black hole—In information terms, a black hole is where information you once had, or knew how to get, goes when you can no longer access it. There may be a lot in the black hole of information. We have no way of knowing. The hole is so disorganized that you can never hope to find any information that gets lost there. (See Dean's desk.)

BPS—bits per second. A measure of the speed of transmission in data communications

buffer—Any portion of your Macintosh's internal RAM that is set aside by the program as a temporary "holding tank" for data going somewhere else, usually from a remote computer on its way to your disk

carrier, carrier signal—The background signal on a channel that is modified in order to carry information. In telecom, as required by the RS-232 specification, the carrier signal is equivalent to a continuous mark signal (1). In a modem, this may be represented by a tone. Inside the computer, this may be represented by a voltage level.


character—An element of a set of symbols, such as a letter or number, or a special symbol such as ?, +, and . The ASCII code defines the set of characters used in data communication between your Macintosh and other computers.

click, clicking [M]—To position the pointer with the mouse and then to press and quickly release the mouse button

Clipboard [M]—The holding place for what was last cut or copied; a buffer area in memory. Information on the Clipboard can be pasted into documents.

CR (carriage return)—ASCII character code 13 produced by the Return (or Enter) key on a keyboard used when entering data into the processor. Informs processor when entry is finished

connect time—The time you spend on line with a data base or other remote computer service. It is usually measured in fractions of a minute.

Ctrl (control)—A key on a keyboard that modifies the action of other keys to provide a shorthand way of issuing commands to a program. Other ways of representing control keys are: ^ J, Control-J, and  J (on the Macintosh).

cursor—A flashing or stationary marker (rectangle or thin line, usually) of light, used to indicate where data is expected to be entered on a video display terminal

checksum—A simple method of error detection or error checking that some terminal software packages use. Checksums are relatively simple to implement but may not detect all types of transmission errors.

CRC (cyclic redundancy check)—An error detection and correction method. This form of error checking is more reliable than simple checksumming or character-oriented parity checking.

cut [M]—To remove something by selecting it and then choosing Cut from a menu. The cut selection is placed on the Clipboard.

DA [M]—See *desk accessory*

DB-9 connector—The Macintosh serial connector (**note:** *not* Macintosh Plus). This connector has two rows of pins, one with four and one with five pins.

DB-25 connector—The standard serial connection. It has two rows of pins, 12 pins and 13 pins wide, in a trapezoidal shell. For most connections, only three to six of the pins are actually used since the regulation of data flow is handled by terminal software rather than control lines.

default—The value that a program assumes in the absence of a value entered by the person using the program

desk accessory [M]—A program that's almost always accessible from the Apple menu selection, for example, a calculator or calendar. Also referred to as a DA.

desktop [M]—The display window showing the status of various documents and applications

dialog box [M]—A box containing a message requesting more information from the user. Sometimes these messages warn that something is about to be done that will cause permanent changes, in which case they may be accompanied by a beep.

dial-up lines—Dial-up lines are the two-wire pairs supplied by the common carriers on the public switched telephone network. These lines are used for communicating between microcomputers, mainframes, and minicomputers using the appropriate modems.

DIF (Data Interchange Format)—A file format created by Software Arts to enable information exchange between its VisiCalc program and other software packages

direct-connect modem—A modem that plugs directly into a telephone jack. Also called a hard-wired modem

DOS (Disk Operating System)—The program that is part of a microcomputer system's software for managing traffic between the CPU (central processor), disk drives, and RAM. DOS is responsible for all activities related to using a disk as a data storage device. May be called differently but will usually have the OS ending.

document [M]—Whatever is created by a Macintosh application program. Information that is entered, modified, viewed, or saved. See *file*

download—To capture and save information sent from another computer. This term is gradually falling out of use. See *upload*

double-click [M]—See *click*

duplex—See *full-duplex transmission*

file—Information or data stored on disk and given a name or number. A file can contain any type of information: program, text, graphics, or a mixture of these. Macintosh files are represented by icons on the finder desktop and are also called documents.

Finder, Finder Desktop [M]—The Macintosh display screen showing the status of disks, programs, and documents

folder [M]—A holder of documents and applications on the desktop. Folders are like subdirectories on some hard disk systems on other micros.

full-duplex transmission—Describes transmission in which data flows in both directions at the same time

half-duplex transmission—Describes transmission in which data can flow in both directions but not at the same time

handshaking—Shorthand term for the ritual of exchanged signals two computers go through to establish the proper connection between them

high codes—Sometimes used to refer to ASCII code numbers above 128, which are used to represent a wide variety of characters, depending on the machine involved

icon [M]—An image that graphically represents an object, a concept, or a message

idle—When modems are connected but not transmitting any data, they send a continuous mark tone. This is called the idle state.

LF (linefeed)—In a printer, advancing the paper one printing line up. On a video display screen, a linefeed moves the cursor down to the next line on the screen. The linefeed character is ASCII 10.

macros—A macro is any sequence of commands or characters that can be invoked or executed by some other, shorter keystroke or menu selection. Macros are useful in terminal programs to cut down the amount of on-line typing required, thus saving connect-time charges.

mainframe—Any large, fast and expensive computer with disk capacities of several hundred megabytes and memories of several thousand kilobytes (RAM)

mark tone—The modem tone that signifies a one bit. A continuous mark tone is sent when a modem is in its idle state. See *space tone*

modem—An abbreviation of “Modulator/Demodulator.” A modem is the box that translates the signals coming from your computer into a form that can be transmitted over standard telephone lines. The modem also translates incoming signals into a form that your computer can understand. Two modems, one for each computer, are needed for any data communications over telephone lines.

modem port [M]—The socket on the back of the Macintosh or Macintosh Plus marked with the telephone icon

Modem7 protocol—An extension of the Xmodem protocol that allows groups of files to be transferred as part of a single exchange. Also known as *Ymodem*.

mouse—A hand-operated device attached to the Macintosh with a cord. The mouse is used to move the cursor around the screen and to enter menu or other choices by pressing its button, called clicking.

null-modem cable—A method of wiring two computers together directly. The main data wires are crossed so that one computer’s input is the other’s output, and vice versa. Sometimes called a crossover cable.

open [M]—To make available. Files and documents are opened in order to work with them.

parallel transmission—A method of transmitting data in which the bits travel down a number of parallel wires (in the Macintosh, 16 at a time). The transmission path is also called a bus.

parity—A method of detecting errors at the byte level. An extra bit is added to the byte being transmitted such that the extra bit, when used in a checksum, adds up to a preagreed upon value.

paste [M]—To place the contents of the Clipboard—whatever was last cut or copied—at the insertion point

polling—The technique employed with multiuser computers to make sure that everybody is served in turn. The computer repeatedly cycles through the entire list of on-line users, asking each if they have anything to send. If they do, the computer deals with that particular request and moves on. Often you won't notice any delay in response time. However, if a large number of other users are on line with the same system at the same time, each polling cycle will take longer, and you may experience some delays.

protocol—An agreement or set of conventions governing the exchange of information

RS-232C—A standard for linking two asynchronous serial devices together. The standard defines the signal characteristics to be found on each pin of a physical connector, but it doesn't say anything about the physical size or shape of the connector. There are many variations in both connector dimensions and the number of actual signals provided.

Scrapbook [M]—A desk accessory in which frequently used text or pictures may be saved.

select [M]—To highlight a portion of text or graphics with the mouse prior to cutting or changing it in some way

serial transmission—Method of transmitting data in which bits travel down one wire, one bit after another

signal—A physical conveyor of data, such as voltage

simplex—Describes a channel through which signals can flow in only one direction at a time

space tone—The modem tone signifying a zero bit. In asynchronous communications, the start bit is represented by a space tone. See *mark tone*

start bit—In asynchronous transmission, the bit that comes just before the data bits to signify to the receiving device that data is about to be received, and that it should begin timing and recording incoming data bits. It is a zero ("space").

stop bit—In asynchronous communications, the bit that comes just after the data bits to signify the end of a character to the receiving system and the return to the idle state. It is the same tone that represents a one (mark).

SYLK—A type of file format developed by Microsoft for the exchange of information between application programs. SYLK stands for symbolic link. Such files are transmitted as text but also contain formatting information.

synchronous transmission—A method of data transmission in which only one central timing device synchronizes the sending and receiving devices. The two devices are then said to be in synch. Synchronous transmission does not require the use of start and stop bits, which are used in asynchronous transmission.

terminal—Any device for sending and receiving computer information

terminal emulation—The idea of emulation is that if a “regular” terminal is unplugged from a system, a “foreign” unit can be plugged in and will immediately perform all the functions of the regular terminal. To accomplish this, a terminal emulation program stores the necessary procedures and control codes needed to make proper use of the codes sent by the remote computer. By properly matching procedures and codes, one terminal can emulate another.

transmission—The transfer of data from one location to another using electrical and electromagnetic signals

upload—To send information to another computer from yours. This term is gradually falling out of use. See *download*

write-only memory (WOM)—A form of memory into which information can be stored but never retrieved. The original prototype, developed by the military, has so far been used to store more than 100 trillion words of surplus federal information. This excess data would have cost the taxpayers more than \$250 billion to store in conventional media. See *black hole*

Xmodem protocol—A popular information exchange protocol intended for micro-to-micro data communication

Bibliography

Inside Macintosh, Vols. I, II, and III. Addison-Wesley, Menlo Park, New York. 1982.

Campbell, Jeremy: *Grammatical Man: Information, Entropy, Language, and Life*. Simon and Schuster, New York. 1982.

Chernicoff, Stephen: *Macintosh Revealed* (two vols.). Hayden Macintosh Library, Berkeley, CA. 1985.

Deasington, R.J: *X.25 Explained: Protocols for Packet Switching Networks*. John Wiley & Sons, New York. 1985.

Drexler, K. Eric: *Engines of Creation: Challenges and Choices of the Last Technological Revolution*. Anchor Press/Doubleday, Garden City. 1986.

Edwards, Paul and Sarah: *Working From Home: Everything You Need to Know About Living and Working Under the Same Roof*. J. P. Tarcher, Inc. Los Angeles. 1985.

Garvin, Andrew P. and Hubert Bermont: *How to Win With Information or Lose Without It*. Bermont Books, Washington. 1980.

Gengle, Dean: *The Netweaver's Sourcebook: A Guide to Micro Networking and Communications*. Addison-Wesley, Reading, MA. 1984.

Kelley, Robert E.: *The Gold-Collar Worker*. Addison-Wesley, Reading, MA. 1985.

Nichols, Elizabeth A., Joseph C. Nichols and Keith R. Musson: *Data Communications for Microcomputers: With Practical Applications and Experiments*. McGraw-Hill, New York. 1982.

Padlipsky, M.A.: *The Elements of Networking Style and Other Essays & Animadversions on the Art of Intercomputer Networking*. Prentice-Hall, Inc., Englewood Cliffs, NJ. 1985.

Ralston, Anthony, and Edwin D. Reilly, Jr.: *Encyclopedia of Computer Science and Engineering, 2nd Ed.* Van Nostrand Reinhold, New York. 1981.

Seyer, Martin D.: *RS-232 Made Easy: Connecting Computers, Printers, Terminals, and Modems*. Prentice-Hall, Inc., Englewood Cliffs, NJ. 1984.

Shapiro, Neil L.: *The IBM PC Connection: Telecommunications for the Home and Office*. McGraw-Hill, New York. 1983.

Silberstein, Judith A., and F. Warren Benton: *Bringing High Tech Home: How to Create a Computer-Based Home Office*. John Wiley & Sons, New York. 1985.

Walden, Jeffrey: *File Formats for Popular PC Software*. John Wiley & Sons, Inc., New York. 1986.

Feedback Sheet

We'd appreciate hearing your feedback, comments, and experiences. You can mail such things to us at our business address:

CommuniTree Group
1047 Sutter Street
San Francisco, CA 94109

1. Are there things included in *MacAccess* that should be corrected or updated in future editions?
2. If you would like to be on our mailing list, fill out and return the following coupon, or some reasonable facsimile thereof.

Thanks.
Dean & Steve

Name _____

Address _____

City _____ State _____ Zip _____

E-Mail and Voice Phone Numbers: _____

Check if applicable:

☐ You may not trade, sell, or give away my name and address. It is for CommuniTree's use only.

☐ Please send me information about your *Personal Media* newsletter

☐ on-line version ☐ print version ☐ both

Index



- Abaton drives, 131
- ACK, 191
- acknowledge, 191
- acoustically coupled. *See* modem
- alternative phone service, 156
 - access numbers for, 157
- American National Standards Institute, 202, 217
- American Standard Code for Information Interchange, 50, 53, 79
 - and EBCDIC, 137, 218
 - character set chart, 219
- analog, 2
 - signals, 31
- ANSI, 202, 217
- answerback message, 64
- Apple II
 - family, 244
 - to Macintosh, 21
 - Hayes modem for, 36
- Apple Personal Modem, 9
 - cable to Macintosh Plus, 29
- AppleTalk, 24, 141–142, 236
- applications, 4
- ASCII. *See* American Standard Code for Information Interchange
- asynchronous serial
 - communication, 29
- asynchronous communication, 56
- attack dialing, 74
- Autopilot, 84, 108
- Avatar PA1000 Turbo, 140
- baud, 32, 53
 - defined, 33
 - setting, 24
- Berkely Macintosh User's Group, 170
- BinHex, 92, 247
 - file extensions and, 176
 - how to obtain, 177
 - how to use, 175
- bit-frame settings, most common, 53
- bit rate, 52
- bit(s), 24–26, 32–33, 57, 58, 67, 76
 - bundle, 174, 175
 - high-order, 218
- bits per character, 53
- Black Box Catalog*, 222
- BLAST, 138
 - as full-duplex protocol, 193
- block of bits, 54, 191
- bps, 32, 33
 - 110, 57
 - 2400, 34, 35
 - setting, 24
- Borland International, 163
- Brothers, Dennis, 42, 43, 195
- buffer, 67
- bundle bit, 174, 175
- bytes, 79
- cables, 20, 39, 178–187
 - crossover, 20–21, 39
 - diagnosing, 184
 - gender changers and, 187
 - ImageWriter, 20
 - ImageWriter to Macintosh, 21
 - Macintosh Plus source, 222
 - null-modem, 20, 21, 29, 178
 - sources for, 222
- call-waiting, 155
- carriage return, 64, 68, 108

- Carrier Detect, 28, 30
- Case Rixon Communications, 202
- Cauzin Systems, Inc., 247
- CCIR, 217
- CCITT, 202, 217
- CD. *See* Carrier Detect
- checksum, 54, 191
- Christensen, Ward, 44, 190
- Christensen/Petersen protocol.
 - See* Xmodem
- Clear to Send, 28, 30
 - wire, 26
- communication
 - accessory supply, 222
 - command languages, 83–112
 - computer, 48
 - full-duplex, 27
 - human, 48
 - parameters and protocols, 47, 48
- CompuServe, 49
- conduits. *See* pipes
- Conference Tree, 98, 236
- Congressional Quarterly*, 235
- connectors, 30
 - crossover, 20–21
 - solderless, 187
- connection port, 59
- connections
 - Apple II with Macintosh, 21
 - Macintosh to NEC
 - Spinwriter, 21
- Consumer's Checkbook, 156
- control-Z, as end of file character, 170
- copyright, 44, 45
- CP/M, 190
- CRC. *See* cyclic redundancy
 - checking
- creator labels, 70
- CR/LF, 64
- crossover cables and connectors, 20–21, 39
- CTS. *See* Clear to Send
- cyclic redundancy checking, 192, 194
- DAFile, 70, 248
- data
 - communication equipment, 27
 - encryption of, 198
 - fork, 19–20, 39, 76
 - integrity, 2
 - security and, 198
 - set, 27
 - terminal equipment, 27
- data base files, 173
- data bits, 57
- Data Communications Standards Handbook*, 202
- Data Interchange Format, 80, 171
- Data Set Ready, 28, 30
- Data Sources Directory*, 221
- Data Terminal Ready, 28, 30
- DB-9 connector, 114
- DCE. *See* data communication equipment
- DEC, 135
 - and VT100 test system, 135
- debugging, on-line, 104
- default settings, 65
- DEL character, 218
- Delayed Characters and Lines, 69
- Delphi, 49
- demodulator(s), 31, 39
- digital, 2
 - signals, 31
- DIF, 80, 171
- direct-cable link, 80
- direct-connect. *See* modem
- disk storage, 165
- Dow Jones Market Manager Plus, 132
- DTE. *See* data terminal equipment
- DTR. *See* Data Terminal Ready
- DSR. *See* Data Set Ready
- duplex. *See* full-duplex
- EBCDIC, 137, 218
 - chart, 219
- EIA, 202

- Electronic Industry Association, standards and
 - RS-232, 24-27, 29, 39
 - RS-422, 24-25, 27, 39, 42
- Encyclopedia of Computer Science and Engineering*, 193
- end of file, 170
- end of transmission, 191
- EOT, 191
- error-checking protocols, 36, 54
- Excel, 79

- Fedit, 174, 175
- FG. *See* Frame Ground
- file(s), 2, 17, 169
 - .wks and wk1, 80
 - 1-2-3, 80
 - address, 88
 - automatic conversion of, 92
 - batch, inTalk and, 106
 - cellular, 173
 - contaminated, 2
 - end of, character, 170
 - error-sensitive, 80
 - font information and, 20
 - formats, 169
 - formatting, 79
 - graphics, 171
 - icon information and, 20
 - import/export charts, 207
 - importing and exporting, 20
 - Macintosh data types, 172-173
 - non-error sensitive, 80
 - parts of, 39
 - service, 88
 - text, 172
 - transfers, 79, 80
 - transfer times, 158
 - transparent transfers, 80
 - translation, 79
 - vanilla text, 173
- flow control, 66
- font
 - fixed pitch, 78
 - proportional, 78
- forks, data and resource, 19-20, 39, 76
- formatting, 19, 79
- format bridges, 79
- Frame Ground, 28, 30
- FreeTerm, 42, 151
- freeware, 43, 44
- full-duplex communication, 27, 61, 63
- full-duplex protocols, 193, 194
- full-screen cursor addressing, 61

- garbage software, 45
- gender changers, 187
- graphics
 - bit-mapped, 172
 - interactive, 108
 - QuickDraw, 172
 - terminal, 236

- hacker, 198
- half-duplex, 61, 68, 69
 - protocols, 64
- handshaking
 - sequence, 47
 - Macintosh-to-Macintosh, 47
 - signals, 26
- high or high-order bit, 55
- Hayes
 - command set, 36-37, 249
 - Smartcom II, 81
 - Smartmodem 300, 53
- Hayes, Dennis C., 36
- HFT. *See* Host File Transfer
- Host File Transfer, 140
- host mode, 106

- IBM
 - 2741 Terminal, 57
 - System/360, 218
- idle state, 56

- information, 5
 - government, 235
 - privacy and, 198
 - proprietary, 198
 - searches, 160
 - services, 223
 - work at home, 234
- inTalk, 148
 - command language and, 83, 105
 - macros and, 89
- Integrated Services Digital Network, 38
- International Association of Cryptologic Research, 237
- International
 - Telecommunications Union, 217
- International Telegraph Alphabet Number 5, 217
- International Consultative Committee on Radio, 217
- International Consultative Committee on Telegraphy and Telephony, 20
- International Standards Organization, 202, 217
- interrupt priority, 24
 - modem port, 24
 - printer port, 24
- ImageWriter
 - cable to Macintosh, 21, 29
- ImageWriter II
 - cable to Macintosh Plus, 21, 29
- ISDN. *See* Integrated Services Digital Network
- ISO, 202, 217
- ITU, 217
- Jazz, 150
- Kermit, 138–140, 193–194
- keyboard emulation, 137
 - EBCDIC and, 137
- LAN. *See* local area network
- Lempereur, Yves, 177, 193, 198, 248
- Life Processor, 203
- linefeed, 64
- local area network, 24
- local echo, 61
- log-on procedure, 47
- Lotus Development, 80
- MacBinary, 195–198
- MacCharlie, 128–131
- MacGeorge/MacMail, 148
- Macintosh
 - cable to ImageWriter, 21
 - Development System, 70
 - family, 240
 - to any machine checklist, 115–116
 - to Apple II, 21, 122–123
 - to DEC as VT100 terminal, 135
 - to IBM PC, 123–132
 - to IBM 3270, 135–136
 - to Macintosh, 113–119
 - to Macintosh through intermediary, 119–121
 - to mainframe services, 131–132
 - to mainframes and minis, 133–141
 - to NEC Spinwriter, 21
 - to PC, 78
 - RS-422 connector for, 28
 - serial connections, 30, 114
- Macintosh Plus
 - cable to ImageWriter II, 21
 - keyboard characters, 138
 - RS-422 connector for, 28
 - serial port connector, 114
- Macintosh Terminal Emulation Program, 43
- MacLink, 78, 121, 124
 - file formats handled, 124
- MacMainframe DA, 140
- macros, 88–90

- MacTEP. *See* Macintosh Terminal Emulation Program
- MacTerm, 152
- MacTerminal, 149
- Mactools, 70
- mainframe, 4
- Mainstay, 200
- mark parity, 56
- mark tone, 56
- MCI, 154
- MCS, 198
- meme, definition of, 176
- message management, 157
- micro, 5
- micro-host, 74
- MicroEditor, 104
- MicroModem II, 36
- MicroPhone, 41, 42, 43, 46–50, 149
 - and Watch Me feature, 75, 102
 - application generator, 102
 - command language, 83
 - customizing, 72
 - FILE** menu, 46
 - FILE TRANSFER** menu, 75, 82
 - PHONE** menu, 49, 72
 - SETTINGS** menu, 49
 - Watch Me feature, 93
- Microsoft, 171, 172
- modem, 2, 30, 31
 - acoustically coupled, 31
 - and ports, 24
 - asynchronous, 188
 - autoanswer/autodial, 93
 - direct-connect, 31
 - Hayes commands, 249
 - Hayes result codes, 249
 - speed of, 32
 - synchronous, 56–57, 188
 - 2400 bps, 34, 35
- modem type
 - Hayes compatible, 58
- MODEM.COM, 190
- modulator(s), 31, 39
- MS DOS family, 240
- Multichannel Communication System, 193, 198
- MultiMate, 79
- MultiPlan, 79, 171
- NASA, 237
- N'Cryptor, 200
- NAK, 191
- National Telecommunications and Information Administration, 235
- NEC Spinwriter, connecting to Macintosh, 21
- nested programs, 105
- networks, 5, 7, 223
- NTIA, 235
- NUL character, 69, 218
- null-modem cable, 20, 29, 30, 39
- packet of bits, 54, 191
- parallel communication, 24–26
- parameters
 - communication, 47
- parity, 54, 56
 - bit, 54
 - even, 55, 58
 - odd, 55, 58
 - mark. *See* mark parity
 - no, 55
 - space. *See* space parity
- password protection, 106
- PC TeleTree, 236
- PC to Mac and Back, 150
- personal media, 203
- Petersen, Keith, 190
- phone, 3
 - line(s), 3, 37
 - noise, 37, 53
 - system, 38
- pins, 30
- pipes, 52
- ports, 22
 - modem, 24, 59
- Pretty Good Terminal, 46, 151

- printer port, 24
- prisoner's ASCII, 51
- protocols, 47, 48, 52, 71
- protocol conversion, 137
- public domain. *See* software

- quick reference sheet, 161 R RAM
 - disk, 24

- RC. *See* Receive Clock
- RD. *See* Receive Data
- Receive Clock, 28
- Receive Data, 28–30, 178
- Red Ryder, 42, 54, 151
 - command language and, 83, 106
- Remote Service Procedures, 84
- Request to Send, 28, 30
- resource fork, 19–20, 76
 - creator labels and, 70
- RI. *See* Ring Indicator
- Ring Indicator, 28
- ringback, 36
- RS-232, 24–26, 39
 - connector, 27
 - lines, 27, 30
 - as a quasi-standard, 29
- RS-422, 24–25, 39, 42
 - connector, 27–28
- RTS. *See* Request to Send

- SCC. *See* Serial Communications
 - Controller
 - script, 72, 73, 75
 - search specialists, 160
 - search statement, 162
 - security checklist, 201
 - serial communication, 24–26
 - Serial Communications
 - Controller, 24
 - serial connections
 - Macintosh, 30
 - serial interface standard, 26
 - serial ports, 22
 - service
 - definition of, 73
 - files, 88
 - settings files, 47
 - SG. *See* Signal Ground
 - shareware, 43, 44
 - Sidekick, 163
 - signal, 2
 - lines, 26
 - Signal Ground, 28, 30
 - SmartCable, 185
 - Smartcom II, 108, 149 *See also*
 - Hayes
 - and command language, 83, 108
 - softspace, 1
 - Softstrip, 177, 247
 - software, 3, 45
 - public domain, 43
 - Software Arts, 79, 172
 - SOH, 191
 - soldering, 186–187
 - solderless connectors, 187
 - space parity, 56
 - space tone, 56
 - special interest networks, 237
 - specialized carriers, 155
 - SPINs, 237
 - standards organizations, 201
 - start bit, 57
 - start of header, 191
 - startup action, 71
 - stop bit, 57
 - one, 57
 - one-half, 57
 - Straight Talk, 150
 - SYLK, 79, 171
 - Symbolic Link Format, 79, 171
 - synchronous communication, 29, 56

 - tabs, 78
 - TC. *See* Transmit Clock
 - TD. *See* Transmit Data
 - Tektronix graphic terminals, 134

- Telebit Trailblazer, 189
 - and inTalk software, 189
 - and MicroPhone software, 189
 - telecom software, 4
- telecom automation, 85–112
- telephone
 - basic service, 154
 - bills, analysis of, 156
 - management, 153–167
 - noisy lines, 96
 - special services and, 154
 - software and, 163
 - wiring and phone jacks, 154
- Telescope, 149
- Teletype, 59, 60
- Telstar Computer Communications*
Catalog, 222
- Tempo, 92
- terminal, 4
 - comparison chart, 144–147
 - families, 244
 - line-oriented, 60
 - programmable, 83
 - software summaries, 148–152
- terminal emulation, 4, 60, 134–138
- Termworks, 46, 152
- text
 - capture, 48
 - file, 49, 77
 - vanilla transfer, 76, 79
- throughput, 11
- TOPS, 142
- top-down design, 99
- Trailblazer. *See* Telebit Trailblazer
- Transmit Clock, 28
- Transmit Data, 28, 29, 30, 178
- TTY. *See* Teletype
- turnkey communications, 72
- UNIX, 121
 - macget*, 121
 - macput*, 121
- US Sprint, 155
- vanilla text, 9, 76–77
 - transfers, 116
- vaporshareware, 43
- variables, string and numeric, 104
- VC format
- VAX/VMS, 121
 - XMAC, 121
- VersaTerm, 134, 150
- VisiCalc, 80, 171
- voltmeter, using, 185
- VT100, 61, 134
- Wait For Echo, 68
- Wait For Prompt, 68
- wait state, 57
- Watch Me, 84
- Watson, Scott, 106, 107
- Winograd, Ken, 248
- Word document
 - as text-only file, 77
- WordStar
 - text formatting and, 126–128
- word wrap, 70
- work at home, 234
- Write a Procedure For Me, 84, 107
- X.PC, 193, 194–195
- X-On/X-Off, 66–69, 218
- Xmodem, 44, 190
 - and MacBinary, 196
 - CRC and, 192
 - drawbacks of, 191–192
 - relaxed, 192
- Ymodem, 190
- Ziff-Davis Publishing Company, 221

MacAccess

Information in Motion

Increase the power and flexibility of your Macintosh by learning to transfer and share data between your computer and another—whether that computer is a Macintosh, an IBM PC, an Apple II, or your company mainframe.

MacAccess explains how to link up your Macintosh to other computers by examining the many hardware solutions. It discusses low-cost options—such as building a simple cable to link your Mac directly to an IBM PC—and expensive solutions like adding a MacCharlie, a Mac/IBM hardware marriage. It also describes the available soft-

ware packages designed to make linking up swift and hassle-free. Advanced topics, such as file conversion, cable wiring, and protocols, are included to help you set up and put your system to work.

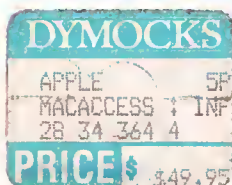
This book provides a clear explanation of telecommunications and discusses specific issues such as proprietary information, data security, and data integrity. It also covers telephone management, including how to better manage information transfer with voice-phone management and online logs, as well as suggesting ways to cut phone costs.



About the Authors

Dean Gengle and Steve Smith are partners in CommuniTree Group, in San Francisco, a firm offering telecommunications consulting and products. Together they have written, edited, designed, and produced numerous computer books and manuals. Dean Gengle

is a telecommunications specialist and has written for a number of magazines including *Dr. Dobb's Journal* and *Computer Dealer*. Steve Smith has contributed graphics support for a number of books including the bestselling *Macintosh Revealed*.



ISBN: 0-672-46567-1



HAYDEN BOOKS

A Division of Howard W. Sams & Company

4300 West 62nd Street

Indianapolis, Indiana 46268 USA



0 81262 46567 7